

Linux

Г Л А З А М И

ХАКЕРА

6-е издание

Михаил Фленов

Безопасность Linux
Оптимизация ОС и сервисов
Атаки хакеров на Linux
Защита сервера от хакеров
Предотвращение возможных атак
Примеры для CentOS и Ubuntu

Михаил Фленов

Linux

Г Л А З А М И

ХАКЕРА

6-е издание

Санкт-Петербург
«БХВ-Петербург»
2021

Оглавление

Предисловие	11
QualitySource	16
Второе издание	17
Третье издание	17
Четвертое издание	18
Пятое издание	18
Шестое издание.....	19
Благодарности.....	19
 Глава 1. Прежде чем начать.....	 21
1.1. Ядро	22
1.2. Дистрибутивы	23
1.2.1. Red Hat Linux	24
1.2.2. CentOS	25
1.2.3. SuSE Linux	25
1.2.4. Debian	25
1.2.5. Ubuntu.....	26
1.2.6. Raspbian.....	26
 Глава 2. Установка и начальная настройка Linux.....	 27
2.1. Подготовка к установке.....	27
2.2. Начало установки	29
2.3. Разбивка диска	30
2.3.1. Файловые системы	32
2.3.2. Ручное создание разделов.....	35
2.4. Выбор пакетов для установки.....	37
2.5. Завершение установки.....	41
2.6. Пароль.....	41
2.7. Первый старт.....	44
2.8. Мы в системе	49
2.9. Подсказки	50
2.10. Основы конфигурирования.....	50
2.10.1. Запрещено то, что не разрешено.....	51
2.10.2. Настройки по умолчанию	51

2.10.3. Пароли по умолчанию.....	52
2.10.4. Безопасность против производительности.....	52
2.10.5. Внимательность.....	53
2.11. Обновление	54
2.12. Устройство Linux: ядро и модули	54
2.13. Установка дополнительных пакетов в Ubuntu	56
2.14. Установка дополнительных пакетов в CentOS.....	58
2.15. Редактирование файлов.....	59
Глава 3. Добро пожаловать в Linux.....	61
3.1. Файловая система	62
3.1.1. Основные команды.....	64
<i>pwd</i>	64
<i>ls</i>	64
<i>cat</i>	65
<i>tac</i>	66
<i>cd</i>	66
<i>cp</i>	66
<i>find</i>	67
<i>grep</i>	69
<i>mkdir</i>	69
<i>rm</i>	70
<i>df</i>	70
<i>mount</i>	70
<i>umount</i>	73
<i>tar</i>	74
<i>rpm</i>	74
<i>which</i>	74
3.1.2. Безопасность файлов.....	75
Дата и время изменения	75
Контрольные суммы.....	76
Что контролировать?.....	77
Замечания по работе с файлами	78
3.1.3. Ссылки.....	79
Жесткие ссылки	79
Символьные ссылки	80
3.2. Загрузка системы	82
3.2.1. Автозагрузка	82
3.2.2. GRUB2.....	84
3.2.3. Интересные настройки загрузки	85
3.3. Регистрация в системе.....	86
3.3.1. Теневые пароли	86
3.3.2. Забытый пароль	88
3.3.3. Модули аутентификации	89
3.3.4. Сложность паролей	90
3.4. Процессы	91
3.4.1. Смена режима.....	91
3.4.2. Остановка процессов.....	93

3.4.3. Просмотр процессов	93
3.4.4. «Зомби»: поиск и устранение	96
3.5. Планирование задач	97
3.5.1. Формирование задания	98
3.5.2. Планировщик задач	99
3.5.3. Безопасность запланированных работ	101
3.6. Настройка сети	102
3.6.1. Адресация	102
3.6.2. Информация о сетевых подключениях	104
3.6.3. Изменение параметров сетевого подключения	105
3.6.4. Утилита ip	106
3.6.5. Базовые настройки сети	107
3.6.6. Протокол IPv6	108
3.7. Работа с модулями ядра	109
3.8. Переменная <i>\$PATH</i>	110
Глава 4. Управление доступом	113
4.1. Права доступа	113
4.1.1. Назначение прав	115
4.1.2. Владелец файла	117
4.1.3. Правила безопасности	117
4.1.4. Права по умолчанию	118
4.1.5. Права доступа к ссылкам	119
4.1.6. Права доступа к ссылкам	120
4.2. Управление группами	122
4.2.1. Добавление группы	122
4.2.2. Редактирование группы	124
4.2.3. Удаление групп	124
4.3. Управление пользователями	124
4.3.1. Файлы и папки нового пользователя	127
4.3.2. Изменение настроек по умолчанию	128
4.3.3. Редактирование пользователя	129
4.3.4. Удаление пользователя	129
4.3.5. Настройка процедуры добавления пользователей	130
4.3.6. Взлом паролей	132
4.4. Типичные ошибки распределения прав	133
4.5. Привилегированные программы	135
4.6. Дополнительные возможности защиты	135
4.7. Защита служб	137
4.7.1. Принцип работы	139
4.7.2. Установка Jail	140
4.7.3. Работа с программой Jail	141
4.8. Получение прав root	143
4.9. Права приложений	144
4.10. Сетевой экран	145
4.10.1. Фильтрация пакетов	147
4.10.2. Параметры фильтрации	148
Протоколы	150

Фильтрация портов	150
Фильтрация адресов	151
Фильтрация нежелательных адресов	152
Фильтрация неверных адресов	152
Фильтрация в Linux	153
4.10.3. Брандмауэр — не панацея	154
4.10.4. Брандмауэр как панацея	154
4.10.5. Конфигурирование брандмауэра	155
4.10.6. Основные возможности <i>iptables</i>	156
4.10.7. Переадресация	159
4.10.8. Утилита <i>firewalld</i>	160
4.10.9. Uncomplicated Firewall: упрощенное управление	160
4.11. Некоторые нюансы работы с брандмауэром	161
4.11.1. Обход сетевого экрана	162
4.11.2. Безопасный Интернет	164
4.11.3. Дополнительная защита	165
4.12. Запрет и разрешение хостов	166
4.13. Советы по конфигурированию брандмауэра	168
4.14. Повышение привилегий	169
Глава 5. Администрирование	175
5.1. Полезные команды для сетевых соединений	175
5.1.1. <i>ping</i>	176
5.1.2. <i>netstat</i>	177
5.1.3. <i>telnet</i>	178
5.1.4. <i>r</i> -команды	180
5.2. Шифрование	181
5.2.1. Программа <i>stunnel</i>	186
5.2.2. Дополнительные возможности OpenSSL	187
5.2.3. Шифрование файлов	188
5.2.4. Туннель глазами хакера	189
5.2.5. Шифрование домашнего каталога	191
5.3. Протокол SSH	193
5.3.1. Конфигурационные файлы	193
5.3.2. Основные параметры конфигурации сервера SSH	194
5.3.3. Параметры доступа к серверу <i>sshd</i>	197
5.3.4. Конфигурирование клиента SSH	197
5.3.5. Пример работы клиента SSH	199
5.3.6. Вход по ключу	199
5.3.7. Защищенная передача данных	201
5.4. Демон <i>inetd/xinetd</i>	202
5.4.1. Конфигурирование <i>xinetd</i>	203
5.4.2. Безопасность	205
5.5. Анализ безопасности дистрибутива	207
Глава 6. В стиле Samba	211
6.1. Конфигурирование Samba	212
6.1.1. Основные настройки	214
6.1.2. Безопасность	215

6.1.3. Сеть.....	217
6.1.4. Замена сервера Windows.....	218
6.1.5. Поддержка WINS и DNS	218
6.1.6. Отображение файлов.....	219
6.2. Описание объектов	219
6.2.1. Пора домой	219
6.2.2. Доменный вход.....	220
6.2.3. Распечатка.....	221
6.2.4. Общий доступ	221
6.2.5. Личные каталоги	222
6.2.6. CD-ROM.....	222
6.3. Управление пользователями.....	223
6.4. Использование Samba.....	225
6.5. Развитие Samba	226

Глава 7. Веб-сервер 227

7.1. Основные настройки	228
7.2. Модули	230
7.3. Права доступа	231
7.4. Создание виртуальных веб-серверов	237
7.5. Еще несколько слов о безопасности	239
7.5.1. Файлы .htaccess.....	239
7.5.2. Файлы паролей	241
7.5.3. Проблемы авторизации.....	242
7.5.4. Обработка на сервере.....	242
7.6. Проще, удобнее, быстрее	243
7.7. Безопасность сценариев	244
7.7.1. Основы безопасности сценариев.....	245
7.7.2. Модуль mod_security	247
7.7.3. Секреты и советы	249
Ограничение сценариев.....	249
Резервные копии	249
7.8. Индексация веб-страниц	250
7.9. Безопасность подключения.....	252

Глава 8. Электронная почта 255

8.1. Настройка sendmail	257
8.2. Безопасность сообщений	260
8.3. Полезные команды	260
8.4. Безопасность sendmail	261
8.4.1. Баннер-болтун.....	261
8.4.2. Только отправка почты.....	261
8.4.3. Права доступа	262
8.4.4. Лишние команды	262
8.4.5. Выполнение внешних команд	263
8.4.6. Доверенные пользователи	263
8.4.7. Отказ от обслуживания.....	263
8.5. Почтовая бомбардировка	264

8.6. Спам	265
8.6.1. Блокировка приема спама.....	265
Фильтрация серверов	265
Фильтрация сообщений.....	266
8.6.2. Блокировка пересылки спама.....	268
8.7. Сервер Postfix.....	270
8.7.1. Псевдонимы	270
8.7.2. Ретрансляция	271
Глава 9. Шлюз в Интернет	273
9.1. Работа прокси-сервера	273
9.2. Кэширование	277
9.3. Прокси-сервер squid	277
9.3.1. Директивы настройки HTTP	278
9.3.2. Директивы настройки FTP	279
9.3.3. Настройка кэша	279
9.3.4. Журналы.....	282
9.3.5. Разделение кэша	282
9.3.6. Дополнительные директивы	283
9.4. Права доступа к squid	284
9.4.1. Список контроля доступа	284
9.4.2. Определение прав.....	286
9.4.3. Аутентификация	287
9.5. Некоторые нюансы работы со squid	288
9.5.1. Безопасность сервиса	288
9.5.2. Ускорение сайта	289
9.5.3. Маленький секрет поля <i>User Agent</i>	289
9.5.4. Защита сети.....	289
9.5.5. Борьба с баннерами и всплывающими окнами.....	290
9.5.6. Подмена баннера	291
9.5.7. Борьба с запрещенными сайтами.....	294
9.5.8. Ограничение канала	295
9.6. Защита прокси-сервера: squidGuard.....	298
9.6.1. Установка	298
9.6.2. Настройка.....	300
9.7. Шлюз в Интернет.....	302
Глава 10. Передача файлов	305
10.1. Протокол FTP.....	306
10.1.1. Команды протокола FTP.....	306
10.1.2. Сообщения сервера	309
10.1.3. Передача файлов	311
10.1.4. Режим канала данных	312
10.2. Сервер ProFTPd.....	313
10.3. Еще несколько слов о протоколе FTP.....	315
Глава 11. DNS-сервер.....	317
11.1. Введение в DNS	318
11.2. Локальный файл hosts	319

11.3. Внешние DNS-серверы	320
11.4. Настройка DNS-сервиса	321
11.5. Файлы описания зон	323
11.6. Обратная зона	325
11.7. Безопасность DNS	326
Глава 12. Мониторинг системы	329
12.1. Автоматизированная проверка безопасности	330
12.2. Закрываем SUID- и SGID-двери	333
12.3. Проверка конфигурации	334
12.4. Журналирование	337
12.4.1. Основные команды	337
<i>who</i>	337
<i>users</i>	338
<i>last</i>	338
<i>history</i>	338
<i>lastlog</i>	339
<i>lsuf</i>	340
12.4.2. Системные текстовые журналы	340
12.4.3. Журнал FTP-сервера	342
12.4.4. Журнал прокси-сервера squid	344
12.4.5. Журнал веб-сервера	345
12.4.6. Кто пишет?	345
12.4.7. Утилита logrotate	350
12.4.8. Пользовательские журналы	353
12.4.9. Обратите внимание!	354
12.5. Работа с журналами	356
12.5.1. Команда <i>tail</i>	357
12.5.2. Программа <i>swatch</i>	358
12.5.3. Программа <i>Logsurfer</i>	358
12.5.4. Программа <i>Logcheck/LogSentry</i>	358
12.6. Безопасность журналов	359
12.7. Мониторинг ресурсов	361
Глава 13. Резервное копирование и восстановление	363
13.1. Основы резервного копирования	363
13.2. Доступность на все 100 процентов	365
13.3. Хранение резервных копий	366
13.4. Политика резервирования	367
13.4.1. Редко, но метко	368
13.4.2. Зачастили... ..	368
13.4.3. Часто, но не все... ..	369
13.4.4. Периодично	369
13.4.5. Полная копия... ..	370
13.5. Резервирование в Linux	370
13.5.1. Копирование	370
13.5.2. Утилита <i>tar</i>	371
13.5.3. Утилита <i>gzip</i>	373
13.5.4. Утилита <i>dump</i>	374

13.6. Защита резервных копий.....	375
13.7. Облака.....	376
Глава 14. Советы на прощанье.....	377
14.1. Пароли	377
14.2. rootkit: «набор администратора»	380
14.3. backdoor: «потайные двери»	383
14.4. Небезопасный NFS	384
14.5. Определение взлома	386
14.5.1. Осведомлен — значит защищен	386
14.5.2. Ловля на живца	388
14.6. Тюнинг ОС Linux.....	389
14.6.1. Параметры ядра	390
14.6.2. Тюнинг HDD.....	393
14.6.3. Автомонтирование	394
14.7. Короткие советы	396
14.7.1. Дефрагментация пакетов	396
14.7.2. Маршрутизация от источника	396
14.7.3. SNMP	397
14.7.4. Полный путь	398
14.7.5. Доверенные хосты.....	399
Заключение.....	401
Приложение 1. Команды протокола FTP	403
Приложение 2. Полезные программы.....	405
Приложение 3. Интернет-ресурсы	407
Приложение 4. Работа в командной строке	409
Псевдонимы	409
Перенаправление	410
Запуск в фоне	410
Последовательность команд	411
Предметный указатель	412

Предисловие

Эта книга посвящена рассмотрению одной из самых популярных операционных систем (ОС), устанавливаемых на серверы, — ОС Linux. А если учесть, что Андроид тоже построен на базе Linux, то и этой мобильной ОС, хотя мы остановимся только на ее серверной версии.

Для домашнего использования ОС Linux за долгие годы своего существования пока еще не получила такой популярности, как среди профессиональных администраторов. На мой взгляд, проблема кроется в графическом интерфейсе и отсутствии необходимых программ. Но это мое личное мнение.

Графические оболочки Linux выполнены в довольно спорном дизайне. Рабочую среду GNOME и ее разработчиков раскритиковал даже сам создатель Linux — Линус Торвальдс.

Я больше предпочитаю классические, простые и строгие цвета и, наверное, поэтому последние два года больше всего времени провожу в macOS, которая является достаточно близким родственником Linux, потому что построена на компонентах BSD (обе ОС имеют UNIX-корни). Ядро, которое используется в macOS, когда-то создавалось программистами BSD, хотя его присутствие в ее дистрибутивах никогда официально не признавалось. Плюс очень много утилит в эту ОС от Apple также пришли из мира UNIX.

Операционная система macOS и различные дистрибутивы Linux очень схожи — много общего в работе консоли, а такие понятия, как конфигурация MySQL, PHP, Apache, вообще идентичны.

Второй недостаток Linux — нехватка игр и бизнес-программ. Опять же, для Windows и macOS существуют MS Office и вся линейка продуктов Adobe, а бесплатные офисные пакеты под Linux пока все еще очень сильно проигрывают MS Office. Да и Adobe Photoshop до сих пор остается бесспорным лидером. Это снова мое личное мнение — кому-то The GIMP может быть удобен и Google Docs достаточно.

Но вот что касается серверных приложений, где не нужно никаких красот, а требуются лишь производительность и надежность, и достаточно просто командной строки или управления через удаленный терминал или браузер, — то тут все пре-

имущества Linux выходят на первый план, и здесь она способна конкурировать с другими ОС. Недаром большинство серверов строят именно на Linux, потому что ее серверные компоненты вполне конкурентоспособны с аналогами из других платформ и даже превосходят их.

Лично я использую Linux главным образом как серверную систему и в основном в Сети. Она бесплатная, и это позволяет сэкономить пользователям и компаниям во всем мире огромные суммы денег на лицензиях. По работе мне приходится иметь дело с серверами как на Windows, так и на Linux, и вторые обходятся намного дешевле, так что для своих проектов я всегда выбираю их.

Установка ОС Linux становится проще, а графический интерфейс и удобство работы в некоторых случаях не уступают самой распространенной на настольных системах операционной системе Windows. Самое главное, что ценят пользователи Linux, — это возможность ее настраивать.

Когда-то Windows 9x и даже Windows XP можно было настраивать на любой вкус и существовали программы, которые изменяли внешний вид рабочего стола до неузнаваемости. В Windows Vista возможности по настройке сильно сократились, а с полным изменением рабочего стола в Windows 8/10 менять стало практически нечего.

За долгие годы существования Linux осталась гибкой. На нее можно устанавливать различные графические оболочки, можно даже сделать так, что рабочий стол станет выглядеть как Windows, как macOS или даже как нечто космическое. Именно за эту гибкость Linux любят во всем мире.

Мы будем рассматривать в этой книге Ubuntu (ее корни — Debian) и немного затронем CentOS. Согласно статистике Интернета, Ubuntu является самой популярной сборкой Linux. Чтобы узнать эту статистику, достаточно зайти в рейтинг **mail.ru** и посмотреть любой популярный открытый сайт. Статистика **mail.ru** показывает, с каких ОС и с каких браузеров заходят на сайт пользователи. Так вот, среди дистрибутивов Linux с огромным отрывом на первом месте находится Ubuntu, поэтому имеет смысл выбрать именно ее.

CentOS сильно отличается от Ubuntu — у нее другой пакетный менеджер, и ее больше сравнивают с Red Hat Linux. Очень часто можно услышать, что CentOS — это бесплатная версия Red Hat Enterprise Linux. Не знаю, насколько верно это утверждение. Этот дистрибутив я чаще вижу на веб-серверах, и мои сайты работают на выделенном хостинге именно под CentOS.

Я не смогу «покрыть» в этой книге обе ОС полностью и не планирую этого делать. Но я постараюсь вас ими заинтересовать и дать такую основу, чтобы дальше вы могли двигаться уже без меня.

В мире Linux — в различных ее дистрибутивах — используется значительное количество общих компонентов. Как я уже говорил ранее, macOS от компании Apple основана на BSD (одном из вариантов UNIX-систем), и в ней также очень много общего с Linux, — я могу без проблем копировать некоторые конфигурационные файлы с macOS на Linux, и они будут работать без изменений.

Почему так вызывающе называется книга? Это маркетинговый ход или просто какой-то трюк? Когда я написал первую книгу по программированию на Delphi, то основой для нее послужили статьи из журнала «Хакер». Тогда редакция и предложила мне название: «Delphi глазами хакера». Нет, это не мои глаза имелись тут в виду, имелся в виду стиль журнала, — ведь я тогда вплотную работал с ним и очень много для него писал. На обложке той книги даже слово «хакер» было набрано в стиле оформления журнала.

Впоследствии эти «глаза» стали использоваться и для других моих книг, хотя вид слова «хакер» на обложках пришлось изменить, — издательство «Gameland» запретило оформлять это слово их шрифтом.

В названии книги не имеется в виду, что хакер — в смысле взломщик — это я. Я как раз к таким никогда себя не относил, и больше предпочитаю создавать и защищать. Но в книге все же будут присутствовать глаза некоего взломщика, который будет смотреть на вашу систему извне. А чтобы понять, как защищаться, нужно понять, как думает тот, от кого вы защищаетесь, и знать, как он может атаковать.

Эта книга посвящена безопасности, но не только ей. В основном мы будем говорить о самой Linux, и моя главная задача — погрузить вас в ее увлекательный и захватывающий мир. Я не стану пытаться описать всю систему, потому что для этого книга должна быть в несколько раз толще, — мы рассмотрим только самое интересное на мой взгляд.

Я интересуюсь взломом и постоянно изучаю новые его методы, но только потому, что хочу строить безопасные системы, и безопасность меня интересует намного больше. Любой объект может быть рассмотрен с разных точек зрения. Простой пример из жизни — нож, являясь столовым прибором, при определенных обстоятельствах становится орудием убийства или средством самообороны. Точно так же программные инструменты, утилиты или даже просто знания могут быть восприняты и как советы для повседневного ухода за ОС, и как способы защиты ее от проникновения или же как средства взлома системы. Я надеюсь, что вы не станете использовать полученные знания в разрушительных целях. Это вас не украсит и славы не добавит. Да и зачем вам нужна «черная» популярность взломщика? Не лучше ли посвятить себя более полезным и добрым вещам?

Причины, по которым люди идут темной дорогой, могут быть разными. Кто знает человека, который разработал средства безопасности для какой-либо крупной компании или сайта? Хотя бы одного из людей, отвечающих за безопасность крупных сайтов? Мало кто слышал о них... Но стоит взломать Facebook или любой другой сайт, как о вас заговорят во всем мире. Стремление к такой дешевой популярности понятно, но оно совершенно бессмысленно и не нужно.

Когда меня спрашивают, что я подразумеваю под словом «хакер», я привожу простейший пример — если вы как администратор установили и заставили работать ОС и вам удалось настроить ее на максимальную производительность и безопасность, то вы — хакер. Умения хакера позволяют создавать что-либо, превосходящее имею-

щиеся аналоги (т. е. более быстрое, удобное и безопасное). Именно такой является сама ОС Linux, созданная хакерами со всего мира.

Для меня слово «хакер» ассоциируется с людьми, которые создают что-то, а не взламывают. И так считаю не только я. В Торонто уже не раз проходил Facebook Hack, на котором люди не взламывали самую популярную пока социальную сеть, а писали код, создавали приложения для Facebook и соревновались в умении программировать.

Наверное, пора более детально поговорить о том, что будет ожидать вас в самой книге.

Для того чтобы установить ОС на домашний компьютер, не требуется много настроек. Большинство операционных систем устанавливаются с настройками по умолчанию вполне безопасно. Простому домашнему компьютеру не нужно открывать никаких сервисов внешнему миру. И только вы сами иницилируете большинство соединений с внешним миром, запрашивая у почтовых серверов свою почту или у веб-серверов интересующие вас страницы, музыку и видео. Поэтому достаточно запретить любые входящие соединения и разрешить только то, что пользователь запросит сам. И если пользователь попросил систему соединить его с «не совсем хорошим» сервером, то это уже будет ошибка пользователя, потому что ОС и программы мало что могут в этом случае сделать.

Единственное, что нужно в качестве дополнительной защиты домашним компьютерам, — антивирусы, которые будут проверять контент, который пользователь получает/скачивает, на предмет возможного вредоносного кода. Пока что в Linux не было серьезных вирусных эпидемий, которые уже не раз захлестывали мир Windows, и большинство пользователей Linux не ставит антивирусов.

В случае с сервером все намного сложнее. Серверы для того и предназначены, чтобы предоставлять пользователям какой-то контент. К ним могут подключаться совершенно разные пользователи со всего мира, которых вы никогда не видели и не знали. Доверие к таким посетителям практически нулевое, потому что никогда не знаешь, кто запрашивает ресурсы сервера, для чего и как он собирается их использовать.

Мне кажется, что большинство недочетов в программах происходит как раз из-за недопонимания этого. Программисты просто не задумывались о потенциальных угрозах и надеялись, что пользователи будут делать только то, что для них предусмотрено. Но хакеры — далеко неординарные пользователи, и они обязательно будут пытаться использовать то, что не должно быть доступно.

В результате настройка сервера превращается иногда в увлекательное приключение, где борются Инь и Янь: разрешить или запретить? Дать возможность или нет?

Для того чтобы правильно настроить сервер, необходимо знать множество параметров, которые большинству пользователей не нужны. Если же просто закрыть глаза и оставить все значения по умолчанию, то об истинной безопасности Linux не может быть и речи. Дело в том, что производитель программы заранее не знает, что

именно нам понадобится, и делает все возможное, чтобы она работала на любой системе, а для этого ему приходится включать в нее много дополнительных возможностей, что делает систему в целом избыточной.

В последнее время разработчики дистрибутивов и других серверных программ стали максимально урезать установки по умолчанию, т. е. разрешать только базовые возможности, а все сетевые сервисы, которые могут позволить хакеру проникнуть на компьютер, отключать. При этом чаще всего производитель предоставляет нам простое и удобное средство для быстрого включения и конфигурирования нужного сервиса.

Если в Windows все делается в визуальных окнах с большим количеством контекстных подсказок, то в Linux большинство настроек осуществляется именно в конфигурационных файлах. Да, для нее тоже существуют утилиты, упрощающие настройку, но функционал не всех их столь удобен и гибок, как простое прямое конфигурирование файлов.

Я всегда рекомендую, изучая какую-либо тему, узнать мнение еще как минимум двух-трех авторов. Так что обязательно возьмите еще несколько других книг. У каждого автора свой подход, и разные темы могут быть раскрыты по-разному. К тому же в книгах часто можно встретить авторские описания собственного их опыта, а это самое важное и дорогое.

Можно, например, почитать работы Дениса Колисниченко, который уже не первый год специализируется в Linux. Я видел одну его книгу лет десять назад, и она мне показалась очень даже интересной. Хотелось прочитать ее полностью, но я уехал в Канаду, а отсюда у меня доступ к литературе на русском языке практически отсутствует. Онлайн-магазины не доставляют книги в Канаду, а качать нелегальные версии я не хочу. Впрочем, вроде бы сейчас издательства в России стали продавать электронные версии книг, и я теперь могу легально покупать книги на русском через Интернет, осталось только найти время на чтение.

Рассматривая Linux, я буду говорить о безопасности и производительности не в отдельных заключительных главах, а практически все время. Из-за этого могут иногда случаться повторы, но я считаю, что это необходимо. Когда человек уже приобрел навыки неэффективной работы с системой, переучиваться ему сложно. Именно поэтому мы будем разбирать последовательно (от азов до сложных вопросов) все аспекты каждой рассматриваемой темы, аккуратно раскладывая полученные знания по полочкам.

Кстати, популярная сейчас на мобильных устройствах ОС Android построена на базе Linux, и появление под нее вредоносного кода показало, что и здесь вопросами безопасности пренебрегать не стоит.

Итак, давайте знакомиться с Linux с точки зрения хакера. Я уверен, что вы посмотрите на нее совершенно другими глазами и найдете для себя много нового и интересного.

QualitySource

Мой взгляд на Linux может вам понравиться, а может и шокировать. Дело в том, что я не принадлежу к сторонникам или поклонникам Open Source, к которому относится Linux. Я отношу себя к сторонникам движения QualitySource (такого реально не существует, это я так его для себя называю), т. е. качественного кода. Мне все равно, какой это код — открытый или закрытый, главное, чтобы он был качественный. Если бы код ОС Windows был открытым, вы бы полезли его смотреть или изменять? Я бы нет, и большинство тоже.

Я даже против изменения исходных кодов. Изменив их под себя один раз, вам придется делать это каждый раз, когда выходит новая версия. Само собой, что в вышедшем завтра обновлении ОС не окажется ваших изменений. Придется снова и снова изменять исходный код или создавать свою личную ветку кода и поддерживать ее самостоятельно. Но если каждый будет заниматься поддержанием своих веток, то когда найти время на собственную жизнь и семью?

Когда только появился Android, то для него все производители начали писать свои оболочки, чтобы выделиться из общего фона. С выходом каждой новой версии этой ОС приходилось ждать, когда оболочки обновят. Это как топтание на месте — постоянно приходится адаптировать и тестировать один и тот же код для новой версии ОС. Всем это явно надоело, и все чаще можно видеть «чистую» ОС Андроид.

Большинство из нас, устанавливая ОС или какую-то программу, хочет, чтобы она стабильно работала и выполняла положенные действия. Какая нам разница, открыт код или нет? Какая нам разница, на каком языке написана программа? Для меня эти вопросы не существенны. Если программа стоит того, чтобы я отдал за нее запрашиваемые деньги, если она достаточно качественна, — то я отдам эти деньги, независимо от того, открыт ли ее код и на каком языке ее написали.

И ОС Linux, и Windows, на мой взгляд, являются качественными проектами, и я использую их одновременно, но для разных задач. Устанавливать сложный, тяжеловесный и дорогой Windows Server ради банального файлового сервера — это глупость, поэтому здесь я использую Linux. Но для сложных финансовых решений я предпочитаю использовать великолепную связку MS Windows и MS SQL Server. Это мое личное предпочтение, которому не обязательно следовать. Вы можете выбрать для управления базой данных связку Linux и MySQL.

С 2009 по 2017 год я работал над сайтами с высокой нагрузкой и в качестве бэкэнда¹ использовал серверы Windows, на которых установлены база данных и веб-серверы (потому что для больших сайтов я все же предпочитаю использовать C#), но на фронтенде² у меня были установлены кэширующие серверы Linux, FTP-серверы на Linux, серверы деплоя (развертывания) и управления также на Linux.

¹ Бэкэнд (от англ. Back-End, оборотная сторона) — программный код, отвечающий за работу с сервером (базой данных), данными (для их дальнейшей записи в БД или отправки клиенту) и т. п.

² Фронтенд (от англ. Front-End, лицевая сторона) — публичная часть сайта, с которой непосредственно контактирует пользователь, и функционал, который обычно обыгрывается на клиентской стороне (в браузере).

К какому миру присоединиться — это личное решение каждого. Единственное, о чем я прошу вас, — не делайте ничего бездумно. Не стоит устанавливать софт только потому, что он относится к Open Source, как и не стоит считать, что коммерческий софт заведомо лучше. Выбирайте своим умом, пробуйте, тестируйте и принимайте самостоятельное решение в зависимости от конкретной ситуации.

Программа не может быть лучше, надежнее или безопаснее других только потому, что у нее открыт код, это бред полнейший. Яркий пример — sendmail. Не очень хороших программ с открытым кодом, как, кстати, и коммерческих, весьма много, поэтому выбирайте за качество, а не за наличие или отсутствие исходных кодов, которые большинству пользователей просто не нужны.

Почему и когда я выбираю Linux? Я программист, который любит Microsoft .NET и PHP. Первый лучше выполнять на Windows-серверах, а второй на Linux. И если для финансовых проектов я выберу .NET за его возможности и простоту разработки, то для любых других проектов я остановлюсь на Linux+Apache+MySQL+PHP (LAMP), как на более экономном варианте. Эта связка не требует дорогих лицензий, отчисляемых Microsoft, а ее серверы будут потреблять меньше памяти.

Сейчас на сервере, где находятся мои сайты, мне выделено всего 512 мегабайт памяти, но монитор показывает, что используются около 300 и есть еще незанятое пространство. На виртуальную машину с таким количеством памяти поставить полноценную Windows не выйдет, — придется ставить только версию Core, для которой 512 Мбайт — минимум, и тут уже о комфортной работе говорить будет сложно. А если нужна Windows с возможностью удаленного подключения по RDP¹, то потребуется минимум 4 гигабайта, а это уже намного дороже.

Второе издание

Чем отличалось второе издание книги? Я бы назвал обе эти книги разными, потому что в новом варианте было приведено намного больше информации. Я переписал абсолютно все и обновил весь текст в соответствии с современными реалиями.

В ходе этой работы были исправлены и некоторые ошибки, присутствовавшие в предыдущем издании. Их было немного, но в Интернете по этому поводу очень красиво писали те, кто почему-то не любит меня и мои книги. Не знаю почему — ведь я никому ничего плохого не сделал... Ошибки есть везде, даже в авторитетных американских изданиях. Просто там новые издания появляются каждый год, поэтому ошибки исправляются достаточно быстро.

Третье издание

В третьем издании я уже в основном обновлял информацию в соответствии с современными реалиями. Компьютерный мир изменяется очень быстро, за что я его и

¹ RDP (Remote Desktop Protocol) — протокол удаленных рабочих столов.

люблю, потому что приходится постоянно изучать что-то новое. Очень много новой информации попало на компакт-диск к этой книге в виде текстовых файлов.

Я хотел донести до читателя как можно больше информации, и при этом не делать книгу слишком толстой и дорогой. Единственный способ сделать это — максимально использовать компакт-диск. Наиболее интересная информация попала в книгу, чтобы ее было увлекательно читать. Некоторые сильно устаревшие участки текста ушли на компакт-диск.

Четвертое издание

Четвертое издание снова было полностью переписано. Я опять пробежался по каждой главе и каждому абзацу и переделал очень многое. Компьютерный мир меняется сильно и быстро. Информация из первого издания уже совершенно не актуальна, и даже то, что я добавлял во втором издании, также сильно изменилось.

В Linux поменялись приоритеты при выборе файлового или почтового сервера по умолчанию. Что остается до сих пор актуальным — так это связка LAMP (Linux, Apache, MySQL, PHP). Она, кажется, будет жить вечно, потому что эти четыре продукта на самом деле весьма качественные и отлично поддерживаются их создателями.

Я даже немного изменил подход к описанию программ и самой ОС. Если раньше я пытался описать как можно больше команд и параметров, то в этом издании я выкинул очень много скучных и лишних описаний и добавил то, что, на мой взгляд, было бы более интересно читателям.

Даже если у вас есть любое из предыдущих изданий, это издание вам так же будет интересно прочесть практически полностью.

И еще одно изменение, произведенное для четвертого издания, — информация, расширяющая и дополняющая материал бумажной книги, теперь не приклеивается к ней в виде компакт-диска, а размещается на FTP-сервере издательства, и электронный архив с этой информацией можно скачать по ссылке **<ftp://ftp.bhv.ru/9785977533331.zip>** или со страницы 4-го издания книги на сайте **www.bhv.ru**.

Пятое издание

Пока еще рано было переписывать всю книгу, потому что в предыдущем издании была проделана большая работа.

Тем не менее в пятом издании я сократил количество вступительного материала и общих слов о Linux, его истории и пр. Все это переехало на мой сайт в раздел статей по адресу: **<http://www.flenov.info/story/category/Linux>**. Взамен я добавил больше практической информации о самой ОС Linux. Девиз этого издания: меньше общих слов — больше дела!

Обновлена информация по управлению сетевым экраном, добавлено немного новых примеров, включая Uncomplicated Firewall¹, описано, как настроить шлюз в Интернет. В главу о работе с почтой добавлен раздел по борьбе со спамом. Приведено больше информации по безопасности — этого много не бывает.

Все предыдущие издания описывали только Ubuntu, а в пятое я добавил информацию про CentOS, потому что это очень распространенный Linux-дистрибутив для серверов.

Шестое издание

Linux очень динамичная система, и новая версия Ubuntu выходит каждые полгода (октябрь и апрель), а каждые два года выходит версия LTS (Long Term Support), которую компания поддерживает после выхода в течение пяти лет. Предыдущая LTS-версия Ubuntu была выпущена в апреле 2018 года, а в апреле 2020-го вышла и ее новая LTS-версия — 20.04. Это издание обновлено с учетом как раз Ubuntu 20.04.

До сих пор я мало уделял внимания дополнительным программам, которые можно использовать для аудита ОС, потому что они не являются частью самой ОС. Но есть утилиты, которые помогают с настройкой ОС, проверяют ее надежность и безопасность, поэтому я решил в этом издании добавить описание дополнительных утилит, и самым ярким новшеством будет Lynis.

Благодарности

В каждой своей книге я стараюсь поблагодарить всех, кто помогал в ее создании и выходе в свет. Без этих людей просто ничего бы не получилось.

Первым делом я хотел бы поблагодарить издательство «БХВ-Петербург», с которым сотрудничаю уже долгие годы. Спасибо руководству издательства, редакторам и корректорам, которые работают со мной и помогают сделать книгу такой, какой я ее задумывал. Ведь писать приходится в тяжелых по срокам условиях, но иначе нельзя — информация может устареть раньше, чем книга попадет на прилавок.

Спасибо всем, кто помогает сделать текст лучше, обложку красивой и книгу доступной всем желающим.

Не устану благодарить родителей, жену и детей за их терпение. После основной работы я прихожу домой и тружусь над очередной книгой. Таким образом, семья может видеть меня только за компьютером, а общаться со мной очень сложно, потому что все мои мысли устремляются далеко в виртуальную реальность.

¹ Uncomplicated Firewall (ufw) — в переводе с англ. «незамысловатый межсетевой экран», обертка для брандмауэра iptables в Ubuntu.

Большая благодарность моим друзьям и знакомым, которые что-то подсказывали, помогали идеями и программами.

Так уж выходит, но в написании каждой книги участвуют и животные. Эта работа тоже не стала исключением. Во время подготовки первого издания книги мой кот Чекист с 23:00 до 1:00 ночи гулял по квартире и просто кричал от скуки. Я не мог уснуть, а значит, больше времени уделял работе.

А самая большая благодарность: вам — за то, что купили книгу, и моим постоянным читателям, с которыми я регулярно общаюсь в моем блоге **www.flenov.info**. Последние мои работы основываются на их вопросах и предложениях. Если у вас появятся какие-либо проблемы, то милости прошу на сайт. Я постараюсь помочь по мере возможности и жду любых комментариев по поводу этой книги. Ваши замечания помогут мне сделать ее лучше.

Все чаще люди предпочитают не покупать книги, а качать из Интернета их пиратские копии, что наносит ущерб и авторам, и издательствам. Доход от книг падает, и многие хорошие авторы перестают писать, ибо настоящему специалисту легко найти достойный источник дохода без лишних мучений. От этого количество хороших книг уменьшается. Боюсь, эта тенденция сохранится.

Если вы нашли в книге какую-нибудь ошибку, просьба сообщить мне об этом через обратную связь на моем сайте **www.flenov.info**. Ошибки могут быть везде, и не только в программах или ОС, но и в текстах книг. Я также жду ваших отзывов о книге и пожеланий, что вы хотите увидеть в ней в будущем, если появится новое издание.

Со мной также можно связаться через Твиттер: @flenov или Телеграмм: @mflenov.

На этом мы завершаем вступительное слово и переходим к наиболее интересной и главной части книги — знакомству с ОС Linux.

Приятного чтения!



ГЛАВА 1

Прежде чем начать...

Много лет назад установка ОС Linux была очень сложной, и далеко не на каждый компьютер систему удавалось установить, — постоянно возникали проблемы с совместимостью, отсутствием драйверов. Впрочем, такие проблемы были даже у Windows 95, которая так же работала с минимальными функциями, пока на нее не поставят необходимые драйверы.

У меня в 1990-х годах были не очень популярный монитор Philips и специфичная видеокарта, которая отсутствовала в списке доступного оборудования, поэтому мне приходилось подбирать подходящие драйверы из тех, что были в наличии для других систем, чтобы рабочий стол отображался на экране в приемлемом разрешении.

Сейчас установка Linux стала настолько простой, что в некоторых дистрибутивах для полной установки надо всего-то несколько раз щелкнуть мышью — все имеющееся оборудование определяется без проблем и устанавливается без необходимости подбирать драйверы.

Если смотреть на ОС Linux с точки зрения пользователя, то после установки системы ничего настраивать не надо, — можно сразу же приступить к работе с любыми офисными приложениями и пользовательскими утилитами. Дистрибутивы для домашних компьютеров очень часто уже включают и офисные пакеты, и все необходимое на все случаи жизни.

Но если речь идет о сетевых и серверных программах, то тут уже необходимы дополнительные действия. По умолчанию в системе должны быть запрещены практически все действия, которые могут привести к нежелательному результату или вторжению по сети. Для изменения ограничений нужно настраивать конфигурационные файлы, редактировать которые крайне неудобно новичку или пользователю, привыкшему работать с окнами, а также использовать специализированные утилиты, большинство из которых имеют интерфейс командной строки.

Из-за этих неудобств мой знакомый администратор Windows-систем сказал: «Linux придумали администраторы, которым нечего делать на работе, для того, чтобы играть с конфигурационными файлами». Смешно... Но я с этим не согласен, поскольку заглядываю в конфигурационные файлы своих серверов очень редко.

Корпорация Microsoft начинала делать свои ОС по принципу «лишь бы было удобно», поэтому когда-то оставалось лишь подключить к ним требуемые компоненты. Но теперь Windows становится с каждым годом все сложнее и безопаснее, а большинство удобных функций, которые могут нарушить защиту, просто отключаются. При необходимости их нужно включать. Начиная с 2008 года эта тенденция приняла совершенно новый и интересный оборот — в Windows Server появилась версия без графического режима. Да, Windows запускается в текстовом режиме, в котором можно полноценно управлять сервером! В Linux все было наоборот — эту ОС создавали с точки зрения «лишь бы было безопасно», а теперь двигаются в сторону наращивания возможностей и упрощения сервисов. Эта тенденция радует не всегда: в некоторых дистрибутивах Linux если установить какой-либо сервис, то инсталлятор мало того, что устанавливает его в максимальной конфигурации, он еще и запускает его автоматически при старте системы. Это очень плохо, и такие вещи нужно пресекать.

Что ж, удобство и безопасность во многом противоречат друг другу, поэтому производителям приходится в чем-то лавировать.

1.1. Ядро

Ядро — это сердце ОС, в котором реализовано управление памятью и другими ресурсами компьютера. Помимо этого, оно позволяет получить доступ к различному «железу». Например, ранние версии ядра обеспечивали работу только двух USB-устройств: клавиатуры и мыши. Современное ядро поддерживает большинство существующих устройств, а дистрибутивы включают драйверы для большинства популярного оборудования.

Обновление ядра позволяет не только получить новые возможности по работе с «железом» и повысить производительность системы, но и исправить некоторые ошибки. Самое главное, что обновление ядра в Linux не влечет за собой переконфигурирования всей ОС, как это происходит в некоторых других системах. Я видел компьютеры, ОС которых были установлены еще несколько лет назад и не перенастраивались с тех пор, — в них только обновлялось ядро и программное обеспечение.

Да что там далеко ходить, я долго пользовался услугами хостинговой компании myhosting, и когда купил у них выделенный хостинг много лет назад, мне досталась CentOS 6-й версии, которая вышла в 2011 году. Я ушел от них в прошлом году, потому что они продолжали использовать эту 6-ю версию. Сейчас снова проверил их сайт — она до сих пор у них стоит, хотя полных обновлений для этой ОС нет с 2017 года, а поддержка закончится в ноябре 2020 года.

Как бы там ни было, но CentOS 2011 года все еще служит множеству клиентов этой хостинговой компании, и пока она не попадала в новостные ленты о массовых взломах сайтов из-за такого старого дистрибутива. А все потому, что ОС обновляет свое ядро и продолжает ставить сама обновления пакетов. От администраторов вмешательства в работу ОС не нужно. Мне только один раз пришлось чинить об-

новление системы, когда что-то случилось, и автоматическое обновление пакетов не сработало, но это отдельная история, которая не касается ядра.

Финальная версия Ubuntu 20.04 содержит ядро 5.4. При подготовке этой книги я использовал бета-версию системы на этом ядре. Сейчас релиз Ubuntu 20.04 уже вышел, и серьезных изменений в ней по сравнению с бета-версией не произошло, — только исправление багов.

Версия ядра Linux 5.4 принесла нам много интересных возможностей, все их описывать здесь смысла нет, но я выделю три самых ярких:

- ❑ поддержка файловой системы ExFAT на уровне ядра. Раньше так сделать было нельзя, потому что патент на эту файловую систему принадлежит Майкрософт, но недавно ее спецификация была открыта. ExFAT — это расширенная версия файловой системы FAT, которая используется еще со времен MS-DOS и до сих пор обеспечивает работу внешних носителей — карт памяти или флеш-дисков, потому что этот формат поддерживается всеми ОС;
- ❑ Fs-verity — это система обеспечения целостности данных и защиты файлов только для чтения, которая поддерживается файловыми системами ext4 и f2fs. fs-verity подсчитывает хэш-сумму и хранит ее в метаданных и в случае обращения к файлу, сумма которого не совпадает со значением в метаданных, выдает ошибку;
- ❑ поддерживается язык Python только третьей версии. На этом языке пишут большое количество скриптов, но при переходе с Python 2 на Python 3 обратная совместимость потерялась, и скрипты 2-й версии могут не запуститься на 3-й. Впрочем, если учесть, что Python 3 появился еще в 2008 году, то убийство его предшественника в 2020-м вполне логично.

Кроме того, улучшена производительность системы и добавлена поддержка нового оборудования.

Если вы собираетесь использовать Ubuntu на домашнем компьютере, то заметите и визуальные изменения — такие как новые значки и даже темный стиль, который сейчас становится популярным во всех ОС.

1.2. Дистрибутивы

В настоящее время существует множество различных дистрибутивов Linux, но между ними легко проглядывается сходство, т. к. большинство имеет общие корни. Например, многие дистрибутивы построены на основе Red Hat Linux. Компании-производители вносят некоторые коррективы в процедуру инсталляции (чаще всего только графические), изменяют список включаемого программного обеспечения и продают под своей маркой. При этом ядро системы и устанавливаемые программы чаще всего поставляются абсолютно без изменений.

Даже если установочные версии имеют разных производителей, в качестве графической оболочки очень часто используются KDE или/i GNOME, а при отсутствии их в поставке эти графические оболочки всегда можно установить дополнительно.

Таким образом, вне зависимости от основного дистрибутива у всех будет одинаковый графический интерфейс.

Еще недавно в Ubuntu можно было встретить Unity — ее графическую оболочку по умолчанию, но от разработки Unity отказались, и дистрибутив снова возвращается к своим корням — к GNOME.

Я долго не мог решить, какой дистрибутив установить, но потом решил выбрать Ubuntu. Предварительно я заглянул на пару сайтов, посвященных Linux, и посмотрел в статистике перечень операционных систем, с которых заходили на сайт пользователи (такую статистику предоставляет счетчик **mail.ru**). Наиболее популярной оказалась Ubuntu. В принципе, зная один дистрибутив, очень легко перейти на другой, ведь каждый из них — та же Linux.

И все же, на мой взгляд, разнообразие дистрибутивов является самым слабым звеном Linux. Когда вы начнете работать с этой операционной системой, то увидите, что большая часть операций не стандартизирована (что можно расценивать как следствие открытости кода). Получается как в поговорке: «Кто в лес, кто по дрова». Это серьезная проблема, которая усложняет восприятие.

На мой взгляд, неудобство от разнообразия дистрибутивов заключается еще и в том, что производителям приходится много топтаться на месте и писать один и тот же код. Лучше бы они объединились и начали быстрее двигаться вперед. Да, страдает конкуренция и возможность выбора, но развитие станет намного эффективнее.

Какой дистрибутив выбрать? Их слишком много, чтобы упоминать все, и я остановлюсь только на самых популярных:

- ☐ Red Hat Enterprise Linux и SUSE — выбираем, если нужна хорошая корпоративная поддержка;
- ☐ CentOS или Debian — чаще выбирают для серверов, когда не нужна дорогая поддержка со стороны производителя;
- ☐ Mint, Ubuntu — хороший выбор для домашнего компьютера, если не нужна поддержка;
- ☐ Kali — прославился хорошим набором для хакеров и специалистов в области безопасности.

А теперь о некоторых из них немного подробнее.

1.2.1. Red Hat Linux

Этот дистрибутив считается классическим и является законодателем моды в развитии ОС. Помимо всего прочего, Red Hat ведет разработку ОС Linux в двух направлениях: для серверных решений и для клиентских компьютеров. Серверный вариант — платный, а клиентский вариант (Fedora) бесплатен и доступен для скачивания с сайта **fedoraproject.org**. Шли разговоры о том, что Fedora будет независимым проектом, но пока он остался под крылом Red Hat.

Дистрибутивы Linux всегда ругают за сложность установки ядра и программ, которые чаще всего поставляются в исходных кодах и требуют компиляции. Компания Red Hat уже давно упростила этот процесс, разработав менеджер пакетов RPM (Redhat Package Manager). Такие пакеты для установки используют и большинство других разработчиков.

1.2.2. CentOS

Этот дистрибутив очень популярен на серверах и у хостинговых компаний. Имя его расшифровывается как Community ENTerprise Operating System. CentOS направлена на корпоративного пользователя, как и ее родительская версия Red Hat Enterprise Linux (RHEL). Оба дистрибутива совместимы между собой, но CentOS распространяется бесплатно.

Поскольку CentOS — бесплатная и очень популярная на серверах операционная система, я буду указывать специфичные для этой ОС команды в случаях, когда они отличаются от Ubuntu.

1.2.3. SuSE Linux

Мне приходилось работать с разными программами от немецких производителей, но удобство работы с ними не просто хромало, а создавалось впечатление, что эти программы — безногие калеки с детства. Однако разработка от SuSE (www.suse.com) опровергает такое мнение. Этот дистрибутив отличается симпатичным интерфейсом и отличной поддержкой оборудования, потому что содержит громадную базу драйверов. Кроме того, программисты SuSE добавили в дистрибутив набор утилит под названием YaST, которые значительно упрощают администрирование.

SuSE Linux — наверное, один из первых дистрибутивов, который стал дружественным к пользователю и применяет большое количество собственных наработок, чтобы сделать жизнь домашнего пользователя проще. Но при этом он всегда был платным, причем цена кусается. Мне кажется, что именно это помешало SuSE занять верхнюю ступень пьедестала и обойти Ubuntu. Сейчас у SuSE Linux есть бесплатная версия — openSUSE, но она пока не набирает популярности.

1.2.4. Debian

Несмотря на то что цель любого производителя — получение прибыли, существует множество дистрибутивов, которые были и остаются некоммерческими. Основным и самым крупным из них можно считать Debian (www.debian.org). Этот продукт создают профессионалы для себя, но пользоваться им может каждый.

ОС Debian имеет больше всего отличий от классической Red Hat, и у вас могут возникнуть проблемы из-за разного расположения некоторых конфигурационных файлов. Но на этом проблемы не заканчиваются. Как и все некоммерческие проекты,

этот дистрибутив сложнее других. Разработчики позиционируют Debian как надежную ОС, и это у них получается, а вот о простых пользователях они заботятся мало, поэтому домашние компьютеры этот дистрибутив завоюет не скоро.

1.2.5. Ubuntu

Это, наверное, один из самых простых дистрибутивов, его поддержку осуществляет компания Canonical Ltd (сайт разработчиков: www.ubuntu.com). При создании дистрибутива Ubuntu основной целью была выбрана простота работы с ним.

Дистрибутив построен на базе Debian. Выбор компании Canonical Ltd ясен — в лице Debian они получили очень безопасную основу. Единственное, что меня настораживает, — большинство статистических обзоров показывает, что ежегодно в Ubuntu находят больше уязвимостей, чем в ее конкурентах. И это притом что в ОС Debian, на которой основана Ubuntu, уязвимостей находят очень мало.

Впрочем, доверять статистикам и аналитикам надо с осторожностью, особенно их выводам по части безопасности, потому что безопасность измерить невозможно. В дистрибутиве может быть найдена сотня уязвимостей, но он останется надежным, если все они незначительные. С другой стороны, достаточно одной уязвимости, но очень серьезной, которой злоумышленники смогут легко воспользоваться, и которую залатают с большим опозданием, чтобы надежность опустилась до нуля.

1.2.6. Raspbian

Это еще одна ОС, построенная на Debian, но ее основное назначение — устройства IoT (Internet of Things, или Интернет вещей). И основное место использования этой ОС — на одноплатном компьютере Raspberry PI.



ГЛАВА 2

Установка и начальная настройка Linux

Установка когда-то была самой сложной процедурой для всех дистрибутивов Linux. Вспоминаются времена, когда нужно было последовательно загружаться с нескольких дискет, а потом следовать сложным инструкциям или самостоятельно набирать команды Linux, которые уже надо было знать.

Еще одна непростая задача — разбиение дисков на разделы. Их нужно иметь как минимум два: основной и раздел подкачки. Проблема в том, что многие боятся манипулировать с дисками, особенно с теми, на которых уже есть информация. И это правильно, потому что известны примеры, связанные со случайной потерей данных.

Во время инсталляции любая ОС должна определить установленное оборудование и подготовить все необходимое для его нормальной работы. В 1990-х годах и в начале 2000-х перечень поддерживаемых дистрибутивом устройств можно было просмотреть за несколько минут, т. к. многие производители игнорировали Linux, не писали необходимые драйверы и при этом не давали нужной информации. Сейчас все намного проще, потому что работа с оборудованием уже лучше стандартизирована.

2.1. Подготовка к установке

Устанавливать желательно самый свежий стабильный дистрибутив, включающий последнюю версию ядра и приложений. Не стоит брать какой-то пыльный диск с полки и ставить систему с него. Во всех программах могут быть ошибки, просто в новой версии о них еще никто не знает и не сможет вас с ходу взломать 😊. Кроме того, надо помнить, что программные средства необходимо своевременно обновлять. Если воспользоваться старым дистрибутивом, то объем обновлений может оказаться слишком большим и занять несколько дней. Не лучше ли установить сразу все новое и максимально быстро запустить сервер в эксплуатацию?

Кроме того, если установить старый дистрибутив и не обновить его до последней версии, то в нем, скорее всего, будут иметься известные хакерам ошибки, а значит,

останется потенциальная угроза взлома вашего сервера. Если этого не сделают сразу, то это может произойти в любой момент. Не стоит считать, что ваш сервер никому не нужен. Даже «пустышка» кому-то может оказаться нужной — например, для рассылки спама или, может, даже для не вполне легальных действий.

Мастера установки для разных дистрибутивов могут различаться, но все они, как правило, имеют схожие окна, и даже последовательность выполняемых действий зачастую одинакова. Дело в том, что программ установки не так уж и много, и большинство разработчиков используют одни и те же программы и не пишут ничего самостоятельно. Разве что оформление отличается.

Итак, приступим к рассмотрению процесса установки. Первое, что нужно сделать, — это определить место, где будет располагаться ОС. Если у вас новый компьютер, жесткий диск не разбит на части и вы будете использовать только Linux, то при установке просто отведете под нее все доступное пространство. Доверьте разбиение системе, и она сделает это вполне оптимально.

Если у вас уже установлена Windows и вы хотите, чтобы на компьютере было сразу две ОС, то придется сделать несколько «телодвижений». Для установки Linux требуется наличие на диске пустого пространства. Нет, это не свободное место на логическом диске C:, а пустота на винчестере, место, не занятое какими-либо разделами. В последних версиях инсталляторов присутствует возможность изменять размер имеющихся на диске разделов прямо в программе установки, причем без потери данных. Раньше при изменении разметки диска имеющаяся на нем информацию пропадала.

Итак, если у вас все пространство диска под разделы уже используется и вы хотите «откусить» небольшую часть этого пространства для установки Linux, то это вполне реально. Данные на существующем диске просто сдвигаются, а в опустошенной части диска может быть создан новый раздел для Linux.

Для повышения надежности процесса сдвига многие рекомендуют сначала произвести дефрагментацию диска. Эта операция вполне безопасна и заключается в том, что разбросанные по всей поверхности диска данные собираются в одном его месте. В процессе работы с файлами данные каждого файла могут быть разбиты на множество кусков и лежать в любых местах диска. После дефрагментации все файлы станут занимать лишь одну непрерывную область. Таким образом, при сокращении размера раздела не придется опустошать от данных его содержимое.

Я не знаю, насколько эта рекомендация еще действительна, — современные программы изменения разделов более качественны.

Прежде чем вставить в новый компьютер компакт-диск или DVD с дистрибутивом и начать установку Linux, желательно определиться с другими системами, — а именно будет ли на компьютере установлена также и Windows. Если да, то я бы порекомендовал сначала установить Windows, и только потом Linux. Дело в том, что установщик от Microsoft затирает загрузочную запись и уничтожает любое присутствие сторонних систем. То есть установленная ранее Linux станет недоступной.

Установщики Linux более доброжелательны к чужим разработкам, и если на компьютере были «окна» от Microsoft, то загрузчик Linux позволит загружать их без

мы рассмотрим критические моменты установки, когда неправильный выбор может повлечь проблемы. И обычно первый из таких моментов — это распределение дискового пространства.

В случае установки серверной версии Ubuntu все шаги будут выполняться в текстовом режиме. Интересно, что в CentOS даже серверная версия устанавливается графической программой установки.

В некоторых дистрибутивах язык устанавливаемой системы придется определять старым способом, т. е. нажимая разные клавиши на клавиатуре. Программа установки при этом определит, какой именно язык используется на компьютере. В графическом режиме обычно предоставляется выбор языка системы. Если пользователь выбрал русский язык, то для уточнения клавиатуры можно будет выбрать какую-либо из разных русских раскладок. Окно выбора языка и раскладки клавиатуры показано на рис. 2.1.

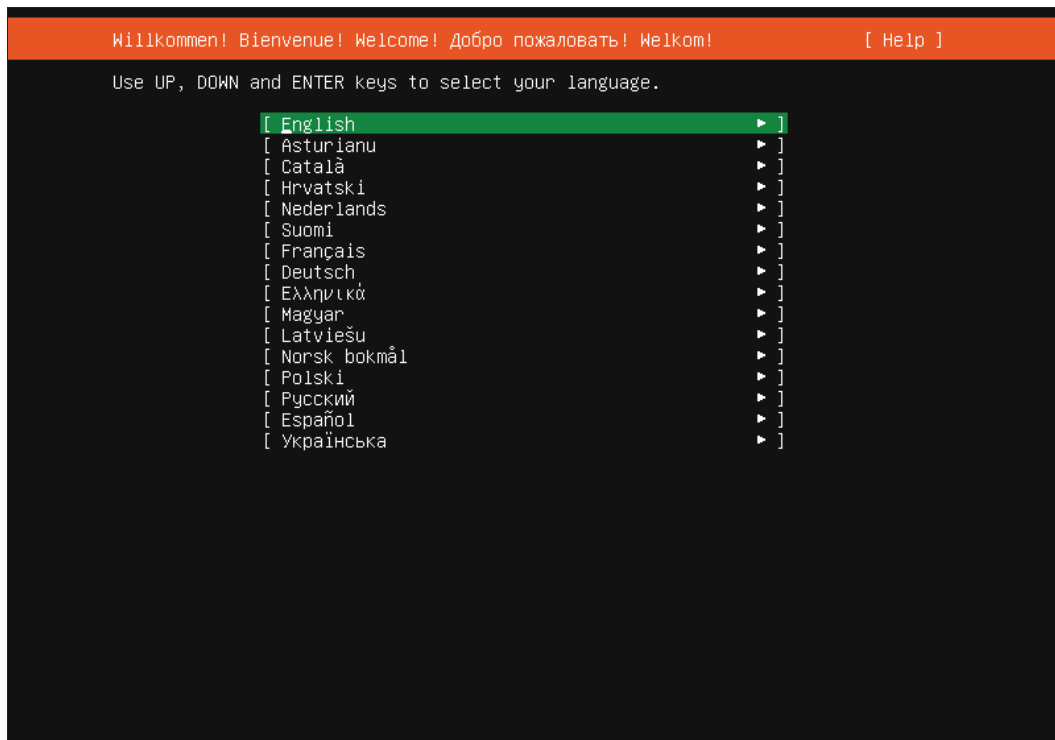


Рис. 2.1. Ubuntu: выбор языка и раскладки клавиатуры

2.3. Разбивка диска

При установке серверной версии Ubuntu шестым шагом идет разбивка диска. Программы установки могут поддерживать три варианта использования дискового пространства для размещения ОС:

Если выбрать вариант **Указать разделы вручную**, вы сможете самостоятельно создать разделы для Linux. В *разд. 2.3.2* мы поговорим о том, какие разделы нужны для работы Linux. В рассматриваемом же нами случае установка идет на пустой диск виртуальной машины, поэтому на нем нет никаких разделов. Если же на вашем компьютере уже есть ОС, то на этом шаге вы можете подкорректировать диск, выделить пустое место из существующего диска или разбить диск так, как вам необходимо.

В Linux диски нумеруются не так, как мы привыкли в Windows. Здесь нет дисков A:, C: и т. д. Все диски имеют имена вида `/dev/hdNX` для дисков IDE и вида `/dev/sdNX` для дисков SCSI. В обоих случаях буква *N* — это номер диска. Например, если у вас два жестких диска типа IDE, то в системе они будут именоваться: `/dev/hda` и `/dev/hdb`.

Что такое *X* в именовании диска? Это номер раздела. Каждый диск может быть разбит на разделы. Пользователи Windows любят создавать два раздела: C: и D:. На первый устанавливается система, а на втором пользователь хранит данные. При такой организации во время переустановки системы можно смело форматировать диск C:, не боясь потерять данные (хотя лучше все же проверить, не попало ли что-то важное случайно на диск C:).

В Linux тоже могут быть разделы внутри одного диска. Например, первый раздел на первом IDE-диске будет иметь имя `/dev/hda1`, второй — `/dev/hda2` и т. д. Цифрой 1 именуется первичный раздел. Под цифрой 2 можно найти расширенный раздел. Логические разделы начинаются с цифры 5.

В выборе конфигурации дисков, показанном на рис. 2.2, есть интересный пункт — **Set up this disk as an LVM group**, который позволяет создать диск в виде группы LVM (Logical Volume Management, логическое управление диском). Работа логической группы чем-то похожа на облачные технологии, при использовании которых вы можете получать ресурсы по требованию, увеличивать мощности при увеличении трафика и сокращать их, если нет надобности платить за лишнее. Точно так же и с дисковым пространством — можно создать логическую группу, которая будет объединять разные диски в одно целое. Самое интересное, что при выборе этой опции можно выбрать для нее шифрование и пароль.

2.3.1. Файловые системы

Теперь поговорим о файловых системах, с которыми работает Linux. От файловой системы зависит качество хранения информации на жестком диске. Linux поддерживает множество систем, в том числе и используемые Windows файловые системы FAT, FAT32 и NTFS, но при установке ОС Linux желательно выбрать родную ей систему: Ext2, Ext3, Ext4¹ или ReiserFS (это название часто сокращают до Reiser). Система ReiserFS является диковиной, потому что используется нечасто,

¹ Сокращение Ext в названии файловых систем расшифровывается как Extended File System (расширенная файловая система).

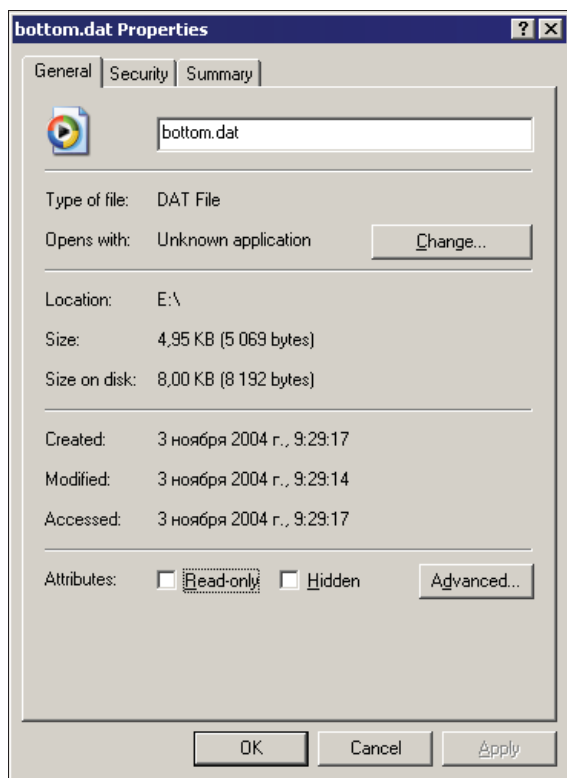


Рис. 2.3. Windows: окно свойств файла

- ❑ Файловая система Ext3 также принадлежит к новому поколению журналируемых систем и работает аналогично ReiserFS. До появления Ext4 она являлась системой по умолчанию в большинстве современных дистрибутивов Linux. Трудно сравнить по производительности ReiserFS и Ext3, но с точки зрения надежности советую использовать последнюю. Разные специалисты придерживаются различных мнений, но я думаю, что стоит согласиться с мнением разработчиков и выбрать Ext3 или ее логическое продолжение Ext4.
- ❑ Четвертое пришествие Extended File System (Ext4) принесло возможность создавать диски размером в 1 эксбибайт (2 в 60-й степени) при размере блока в 4 килобайта. Но более интересным, на мой взгляд, является механизм, который позволяет заранее выделить место под файл и дописывать новые данные в конец уже существующего. Таким образом уменьшается фрагментация данных и повышается производительность файловой системы.

Файловая система Ext4, как уже было отмечено, построена на базе своего предшественника Ext3 и не принесла никаких революционных новшеств, а только улучшение и увеличение показателей предшественницы. Но и этого оказалось достаточно, чтобы Ext4 начали применять по умолчанию в большом количестве дистрибутивов. Мне кажется, это сейчас самая популярная файловая система.

Создание раздела Swap при установке системы в последних дистрибутивах Linux может и не быть предусмотрено, кроме того, вместо раздела подкачки может создаваться файл подкачки. Но если раздел Swap создать можно и вы хотите создать его вручную, то он должен иметь размер не меньше объема доступной оперативной памяти. Сейчас жесткие диски относительно дешевые, и я советую сделать раздел Swap размером в три раза больше, чем установленный на компьютере объем ОЗУ, потому что оперативной памяти может в ряде случаев не хватать, и, чтобы не иметь проблем с разделом подкачки, лучше заранее иметь достаточный запас.

На первый взгляд все это сложно, особенно если вы до этого работали только с Windows. Возможность создания многочисленных папок ОС в виде отдельных разделов является действительно мощной возможностью. Двух разделов достаточно, когда у вас только один жесткий диск и компьютер используется в качестве домашней системы. Если вы используете два винчестера, то лучше создать три раздела:

- ☐ / — на первом диске для всех системных файлов;
- ☐ Swap — на первом диске. Он будет использоваться при нехватке оперативной памяти (современные дистрибутивы могут не требовать этого раздела);
- ☐ /home — на втором диске для хранения пользовательских файлов.

Диски желательно подключить к разным контроллерам (подсоединить на различные шлейфы), что позволит системе работать с устройствами практически параллельно. Это может значительно повысить производительность ОС, потому что Linux сможет работать и с системными, и с пользовательскими файлами одновременно.

Если вы настраиваете сервер, то на отдельные (физические) жесткие диски имеет смысл вынести папки файлов (раздел /home) и раздел /var. Логические диски/разделы в этом случае не дадут желаемого результата. Если же разделы /var и /home вынесены на отдельные диски, то для корневого каталога будет достаточно 3 Гбайт, но можно сделать его и больше.

На разделе /var тоже лучше не экономить и выделить ему 10 Гбайт. Здесь будут храниться файлы журналов, WWW- и FTP-файлы, которые быстро увеличиваются, и если они заполнят все доступное пространство, то система может выйти из строя или просто станет недоступной. Именно на это иногда рассчитывают хакеры, когда организуют атаку «Отказ от обслуживания» (DoS). К этому вопросу мы еще не раз вернемся. Некоторые специалисты по безопасности рекомендуют располагать этот раздел на самом большом диске (чаще всего там же содержится и раздел /home), но это ударит по производительности. Когда журналы находятся на отдельном диске, то это позволяет выполнять запись в них параллельно с обслуживанием остальных разделов. Это значит, что пользователь работает со своими файлами в разделе /home на одном жестком диске, а другой винчестер в это время сохраняет всю информацию об активности пользователя. Если обе папки будут на одном диске, то запись не сможет быть параллельной.

делаем свой компьютер «проходным двором». А если все это активизируется при старте системы, то загрузка компьютера замедлится: в ОС будет открыто множество портов, и заработают разнообразные сервисы, в которых мы пока еще даже не разбирались. А ведь нет идеальных программ — везде существуют ошибки, которые регулярно обнаруживаются и исправляются. И если хотя бы в одном демоне¹ найдется погрешность, то какой-нибудь хакер сможет проникнуть в вашу систему.

Доустановить нужные компоненты после завершения установки ОС стало настолько просто, что я не вижу смысла пытаться ставить сразу все, — добавить недостающие компоненты можно будет в любой момент.

Сразу после выбора серверных компонентов, которые действительно необходимы, программа установки может спросить пароль по умолчанию. Я все не проверял, но знаю, что Ubuntu предлагает установить пароль администратора для MySQL — популярной базы данных уже во время установки. Раньше все компоненты ставились с паролями по умолчанию или вообще без паролей, что тоже не является безопасным.

Старые установщики Linux не предоставляли возможности выбора того, что будет запускаться при старте системы, а банально запускали все сервисы, предполагая, что если мы что-то ставим, то это обязательно нужно запустить. Но в большинстве случаев это не так, потому что мы часто устанавливаем что-то на будущее. Большинство современных дистрибутивов, которые я видел (хотя я видел их не так и много, если учесть, сколько всего доступно), уже предоставляют возможность выбора или не запускают все установленное по умолчанию, и это правильно. Если вы все же решили установить что-то не очень нужное, то стоит обдумать возможность хотя бы запретить его автозапуск.

Вообще, после установки ОС следует проверить, что установлено в автозапуске. Если вы установили что-то на будущее, но не будете сразу же это использовать, то стоит сразу по завершении установки проверить, что запускается автоматом, и отключить запуск ненужного.

В целях упрощения дистрибутивов для домашних пользователей программы установки предоставляют предопределенные варианты установки. Они могут иметь такие или схожие названия:

- ☐ **Быстрая** — использовать определенную производителем конфигурацию. Так вы сократите количество вопросов, на которые надо ответить во время установки, но результат будет далек от оптимального или безопасного;
- ☐ **Разработка** — задействовать основные пакеты и все необходимые компоненты для разработки приложений, компиляции ядра ОС Linux и т. д.;
- ☐ **Офис** — установить типовые пакеты плюс офисные приложения;
- ☐ **Клиент** — включить в устанавливаемую версию ОС только клиентские программы;

¹ Демон — серверная программа, которая обрабатывает запросы клиента.

Если же на взломанном компьютере работает только веб-сервер, то доступ через FTP к конфиденциальным данным хакер получить не сможет. Доступно будет только то, что находится на веб-сервере, а я надеюсь, что там ничего конфиденциального не окажется. Максимум, что сможет сделать хакер, — дефейс (замена главной страницы) или уничтожение сайта. Впрочем, хакер в ряде случаев может получить доступ к базе данных, но это отдельная история.

Пример с веб-сервером не вполне нагляден, потому что этот сервис умеет делать многое. Если хакер получит возможность закачивать свои файлы на сервер, то он сможет загрузить для себя туда Web-shell¹ и выполнять команды и качивать/скачивать оттуда через сеть.

Чтобы злоумышленник не смог, взломав один компьютер, проникнуть на другой, вы должны задавать для каждого из них разные пароли. Некоторые администраторы ленятся запоминать много сложных комбинаций, поэтому придумывают только один пароль и потом устанавливают его везде, где только можно. О паролях мы поговорим в *разд. 2.6* и *главе 3*, но уже сейчас вы должны знать, что для каждой системы должен быть свой код доступа, а пароли сетевых сервисов должны быть максимально сложными.

Но не только демоны являются потенциальной проблемой. В состав Linux многие программы включаются в исходных кодах и должны компилироваться в машинные перед выполнением. Программы, использующие уязвимости Linux-систем, также часто поставляются в исходниках. В самой же ОС Linux очень редко используются инсталляторы программ, поэтому все настройки производятся во время компиляции исходного кода с помощью библиотек разработчика и компилятора gcc. Для того чтобы ими воспользоваться, злоумышленник закачивает модуль в исходниках на сервер, компилирует его и выполняет программу. Чтобы компиляция стала невозможной, я рекомендую не устанавливать библиотеки разработчика и компилятор gcc. Если надо, все можно загрузить уже в скомпилированном виде. Это мелочь, но из мелочей строится вся наша жизнь.

И если gcc окажется взломщику недоступен, то у него возникнут проблемы с выполнением вредного кода. Его ему нужно будет скомпилировать заранее под процессор и ОС вашего сервера, и только потом закачать для выполнения. Не каждый хакер сможет это сделать, потому что большинство их — это скрипт-кидди², которые сидят в Windows.

В последнее время все большую популярность набирают программы на языке Python. На нем пишут веб-сайты, скрипты автоматизации, а также и скрипты для автоматизации взлома или эксплуатации уязвимостей. Однако из-за высокой популярности Python отказаться от его установки не всегда представляется возможным.

¹ Web-shell — это небольшая программка, которую хакер заливает на взломанный веб-сервер, чтобы иметь возможность его удаленно изучать и администрировать.

² Скрипт-кидди (от англ. Script kiddie, скриптовые детки) — в хакерской культуре унижительный термин, используемый для описания тех, кто пользуется для атаки компьютерных систем и сетей скриптами или программами, разработанными другими, не понимая механизма их действия.

Profile setup [Help]

Enter the username and password you will use to log in to the system. You can configure SSH access on the next screen but a password is still needed for sudo.

Your name: mfclubnov

Your server's name: mfclubuntu
The name it uses when it talks to other computers.

Pick a username: mfclubnov

Choose a password: ****

Confirm your password: ****

[Done]

Рис. 2.4. Ubuntu: установка пароля и имени сервера

сколько пользователь `root` имеет полные права на систему, то пароль должен быть как можно более сложным, чтобы его нельзя было быстро подобрать.

Проверка длины пароля — это уже хорошо, ведь это значит, что у администратора просто обязан быть пароль, и «пустой» вариант невозможен. Все специалисты по компьютерной безопасности в один голос просят пользователей не задавать простые пароли, но мало кто следует этим рекомендациям. Нельзя для этих целей использовать имена свои, членов семьи и домашних любимцев, читаемые слова или даты рождения. Такие пароли легко взламываются простым перебором по словарю, и это не отнимет много времени, если словарь хорошо составлен.

А если систему будет взламывать человек, который вас знает, то он сможет обойтись и без словаря и завершит ручной подбор за несколько минут. В прочих же случаях достаточно только собрать данные о вас из открытых источников, таких как Facebook, и записать все найденное в словарь перебора.

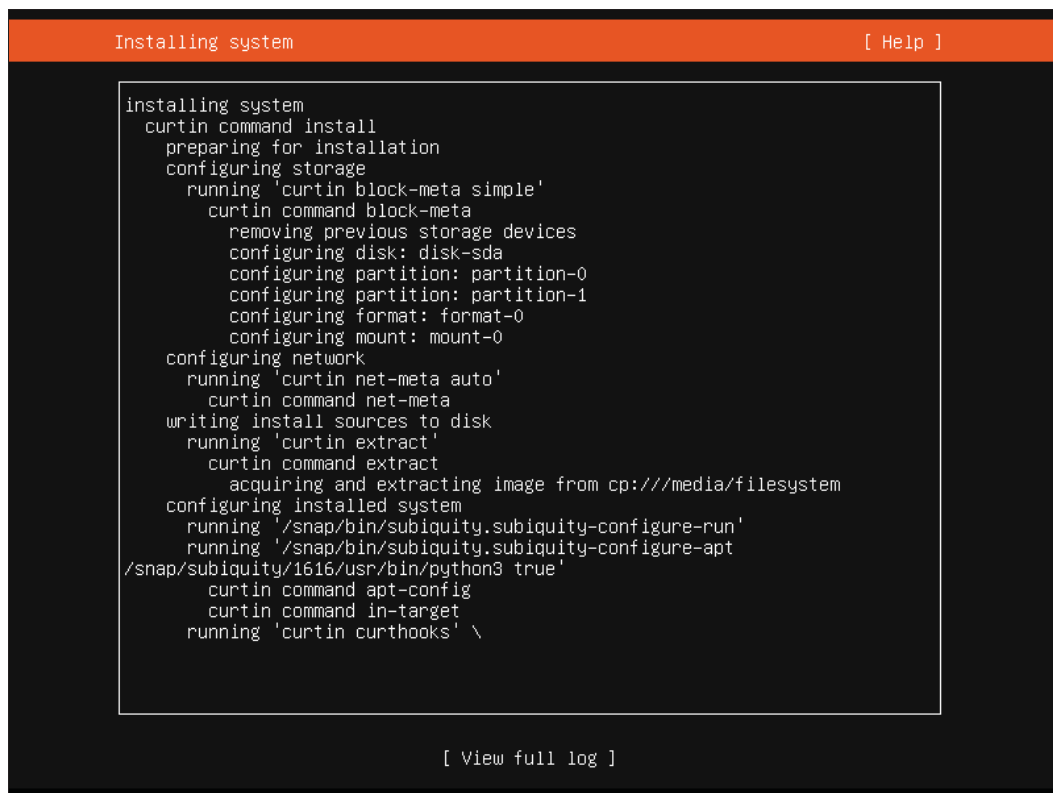
Ну, сейчас некоторые скажут, что вот он — писатель-очевидность, и он тоже будет рассказывать о необходимости выбирать сложные пароли. Да, это очевидно, и я надеюсь, что вы это понимаете, поэтому не буду рассказывать про необходимость сложных паролей, а просто расскажу, как я их выбираю.

При создании пароля желательно генерировать случайные шифры, состоящие из символов разного регистра (прописных и строчных букв), а также цифр и различ-

файлы, но не сможет сделать ничего серьезного с системой, потому что для этого все же нужно знать пароль администратора. Но при наличии в системе уязвимости у хакера может появиться возможность повысить свои права, так что даже от имени простого пользователя автоматический вход лучше не разрешать.

В последних версиях Linux процесс переключения выглядит примерно так же, как и в Windows или macOS, — вы работаете под простой учетной записью, а когда вам нужно выполнить привилегированную операцию, требующую повышенных прав, система запрашивает пароль администратора. Вводя пароль, вы разрешаете системе выполнить эту операцию.

Ну и когда пароли указаны, начинается установка Linux (рис. 2.5).



```
Installing system [ Help ]

installing system
  curtin command install
    preparing for installation
    configuring storage
      running 'curtin block-meta simple'
      curtin command block-meta
        removing previous storage devices
        configuring disk: disk-sda
        configuring partition: partition-0
        configuring partition: partition-1
        configuring format: format-0
        configuring mount: mount-0
    configuring network
      running 'curtin net-meta auto'
      curtin command net-meta
  writing install sources to disk
    running 'curtin extract'
    curtin command extract
      acquiring and extracting image from cp:///media/filesystem
  configuring installed system
    running '/snap/bin/subiquity.subiquity-configure-run'
    running '/snap/bin/subiquity.subiquity-configure-apt'
    /snap/subiquity/1616/usr/bin/python3 true'
  curtin command apt-config
  curtin command in-target
  running 'curtin curthooks' \

[ View full log ]
```

Рис. 2.5. Ubuntu: идет установка системы

2.7. Первый старт

Современные версии загрузчиков красивы и отображают приятное окно выбора системы для загрузки. Просто выберите нужный вариант и откиньтесь на спинку стула — мы погружаемся в мир пингвинов.

только предупреждали об опасности работы под root, то теперь прямо запрещают вход.

Работа под учетной записью root может облегчить взлом компьютера через простые на первый взгляд приложения и способствует вирусным эпидемиям. До сих пор в UNIX-подобных системах почти не было вирусных эпидемий как раз из-за того, что пользователи системы не обладают полными правами. Все программы, которые запускаются пользователями, обладают только теми правами, которые есть у самого пользователя. И чем меньше у вас прав, тем лучше, как бы странно это ни звучало.

Вирусы не могут прописываться в системные файлы, потому что для этого выполняемой программе нужны повышенные привилегии, которые могут быть достигнуты, только если пользователь введет пароль администратора при запуске программы.

Допустим, что вы путешествуете по веб-сайтам. Запущенный вами браузер автоматически имеет те же права, что и вы. И если в нем найдется уязвимость, позволяющая получить доступ к жесткому диску, то злоумышленник сможет воспользоваться этой лазейкой и, например, стереть на нем всю информацию.

Если же работать под пользовательской учетной записью, то уничтожить он сможет только те файлы, которые доступны этому пользователю. Системные файлы в таком случае будут в безопасности. При необходимости повышения прав вы всегда можете это сделать, зная пароль администратора. Для этого служит команда `su`, а в качестве параметра (указывается через пробел после команды) передается имя пользователя, статус которого вы хотите обрести.

Чтобы получить права администратора, наберите команду:

```
su root
```

или

```
su -
```

Вторая команда тоже позволяет приобрести права root, хотя вместо имени указан дефис. В ответ на это система запросит пароль. Введите данные пользователя, права которого вы хотите получить, и после этого вы сможете выполнять команды, доступные этой учетной записи. Например, получив статус администратора, вы приобретете все права на систему.

ПРИМЕЧАНИЕ

Необходимо заметить, что некоторые команды, которые будут описываться в книге, могут оказаться недоступными от имени простого пользователя. Так как мы рассматриваем настройки системы, то некоторые действия могут оказаться опасными, поэтому доступны они только администратору. Если при попытке выполнить что-либо произошло исключение — например, получено сообщение **Команда не найдена**, попробуйте переключиться на root, и, скорее всего, это поможет. Исключением может быть наличие у вас экзотической версии Linux или если вы не установили вызываемую программу.

По правилам безопасности переключение на учетную запись администратора в интерактивном режиме может быть запрещено. Например, если ваше окружение ра-

Прежде чем двигаться дальше и выполнять команды, нужно познакомиться с терминалом. Это командная строка, с помощью которой выполняется управление системой. В серверных дистрибутивах нет графических оболочек (хотя их можно установить), и там управление происходит только в командной строке. В домашних дистрибутивах терминал открывается в виде окна с приглашением вводить команды.

В обоих случаях приглашение вводить команды выглядит следующим образом:

[root@Flenov:/etc]:

Что это значит? Сначала идет имя пользователя, под которым вы вошли в систему, — здесь это **root**. Нет, я реально не вошел под этим пользователем, я его просто показал здесь в качестве примера. После знака **@** следуют имя компьютера и пробел или двоеточие, за которым стоит имя текущей папки. Таким образом, вы всегда знаете, где находитесь и где будут по умолчанию выполняться вводимые команды.

Зачем указывать в командной строке имя компьютера? — ведь мы знаем, за каким компьютером сидим. Ответ на этот вопрос кроется в сетевых возможностях ОС. Вы можете подключиться к удаленному компьютеру с помощью команды `ssh`, и тогда ваша командная строка будет выглядеть совершенно идентично, но вместо локального имени компьютера вы увидите имя удаленного. То же самое я говорил немного `hfytt` в отношении авторизации.

Например, если я из программы терминала выполню команду:

```
ssh flenov@mywebsite.com
```

то откроется сессия с удаленной виртуальной машиной, на которой установлены мои сайты, и я смогу управлять ими, выполняя команды в терминале. То есть, выполняя команды в терминале на моем компьютере, я реально запускаю их на сервере.

На рис. 2.6 показано, как я использую возможности открытия нескольких терминалов. Правда, делаю это здесь из macOS, терминал которого выглядит как окно с вкладками.

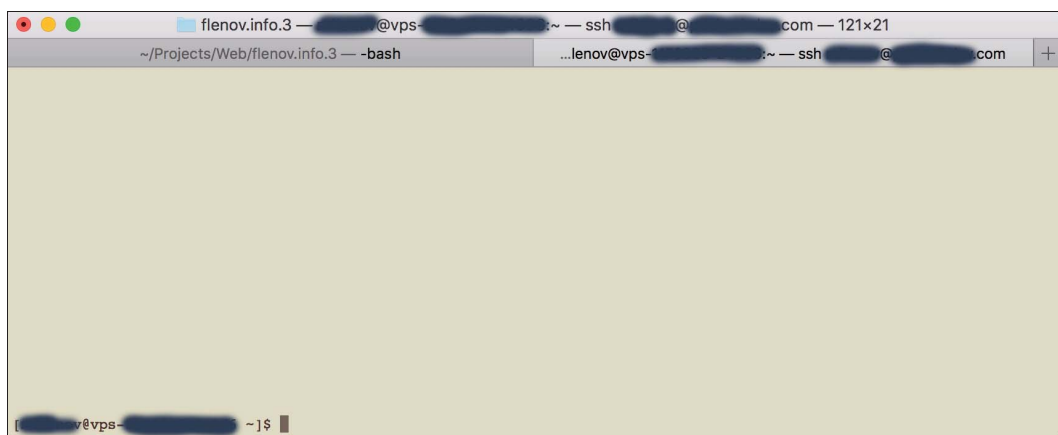


Рис. 2.6. macOS: удаленная сессия `ssh` с виртуальной машиной Linux

Для немедленного выключения компьютера следует ввести:

```
shutdown -h now
```

В последних версиях эти команды выполняются только с правами администратора — команда, отданная от имени простого пользователя, не будет выполнена.

Если же выключить питание с помощью кнопки на системном блоке, то можно потерять данные, которые были загружены в оперативную память, но не сохранены на жестком диске. Это относится к любой ОС — Linux здесь не исключение.

Если вы работаете в текстовом режиме и просто хотите войти под другим именем пользователя, наберите команду `exit`.

2.9. Подсказки

Некоторые операционные системы славятся своей простотой и хорошими подсказками. Но, как мы уже говорили, производительность, надежность и простота — не всегда совместимые вещи. ОС Linux содержит множество команд, и у каждой из них большое количество параметров. Помнить их все невозможно, поэтому разработчики постарались и снабдили ОС обширной справкой.

Если вы хотите получить информацию о какой-либо команде или программе, то попробуйте запустить ее с одним из ключей: `-h`, `-help` или `-?`. Разные программы используют разные ключи, но один из этих, скорее всего, заставит вывести на экран короткую подсказку по использованию программы.

Для получения более подробной информации нужно воспользоваться следующей конструкцией:

```
man имя
```

Здесь *имя* — это название команды или программы, описание которой вас интересует. Например, чтобы увидеть описание команды `shutdown`, выполните:

```
man shutdown
```

Для выхода из программы помощи нужно нажать на клавиатуре клавиши `<->` и `<e>`. В ответ на это перед вами появится сообщение: **Quit at end-of-file (press RETURN)**. Нажмите клавишу `<Enter>`, затем перейдите в конец просматриваемого файла помощи, используя клавиши `<↓>` или `<Page Down>`, и программа `man` завершит работу.

Более простой способ завершения программы `man` — нажать клавишу `<q>`. В этом случае выход произойдет мгновенно.

2.10. Основы конфигурирования

Прежде чем переходить к более глубокому рассмотрению вопросов, связанных с конфигурированием ОС Linux, мы должны определиться с правилами, которые действуют вне зависимости от типа ОС и конкретного сервиса. Придерживаясь их,

Когда вы пишете имя параметра или путь в файловой системе, будьте внимательны не только к словам, но и к их написанию. ОС Linux чувствительна к регистру имен файлов и каталогов. Эта особенность имеет значение и для некоторых конфигурационных файлов.

2.10.3. Пароли по умолчанию

Многие сервисы во время установки прописывают пароли по умолчанию. Раньше в ОС Linux эта проблема стояла очень даже серьезно, потому что установочные пакеты программ не общались с пользователем и не могли попросить его задать пароль. Если это программа, то пароль можно спросить при первом старте, потому что тут возникнет хоть какой-то визуальный контакт с пользователем. А если это сервис, который запускается системой и не имеет никакого способа запросить у пользователя что-то ввести?

Например, база данных MySQL после установки использует для администратора учетную запись без пароля и с именем root. Это имя может ввести в заблуждение, но вы должны знать, что это имя (логин) никакого отношения к системной записи root не имеет. Это внутренняя учетная запись базы данных, и пароли могут и должны быть разными. Сразу после установки MySQL необходимо сменить пароль доступа.

Конфигурированием обычно занимаются программисты, которые настраивают базы под себя и почему-то любят использовать пароли по умолчанию. Я сам программист, и при разработке баз данных тоже иногда так делаю в надежде, что за паролями проследит администратор, а тот надеется на меня, и получается, что мы оба не делаем важных изменений.

Пароли по умолчанию используются не только в программах и ОС, но и в сетевых устройствах, таких как маршрутизаторы и управляемые коммутаторы. В эти устройства встроена система защиты и авторизации. Однако практически все они при производстве снабжаются простыми именами и паролями — типа admin без пароля, или имя и пароль в них устанавливаются одинаковыми.

В Интернете уже давно существуют списки паролей по умолчанию для различных устройств, поэтому не забывайте их менять после установки оборудования.

2.10.4. Безопасность против производительности

Я уже говорил, что безопасность и производительность преследуют совершенно разные цели, которые чаще всего конфликтуют между собой. Настраивая сервер на максимальную безопасность, приходится включать такие сервисы, как журналирование и сетевые экраны, а они расходуют ресурсы процессора и занимают место на диске. И чем больше дополнительных служб включено, тем больше лишних затрат.

Каждый сервис может быть настроен по-своему. Например, в режиме журналирования можно записывать в журналы только основную информацию, что позволит уменьшить нагрузку, но увеличит вероятность того, что какая-то атака пройдет незамеченной.

И в таких случаях я не доверяю своему опыту. Я пишу команды, которые нужно выполнить, зову кого-нибудь из программистов, который будет стоять рядом и следить за отдаваемыми мной командами, и мы вдвоем выполняем все. Опыт — это хорошо, но вот ошибку может совершить каждый.

Когда какой-либо сервер дал сбой и нужно конфигурировать его на ходу, то мы так же работаем над конфигурированием по двое. В этом ничего страшного нет. Помощь никогда лишней не бывает, даже после 25-летнего стажа работы в ИТ.

Графические утилиты более безопасны, потому что каждое действие проверяется и контролируется программой, чего не может сделать текстовая командная строка. Графические программы могут быть удобнее и нагляднее (хотя и не всегда являются таковыми, потому что это зависит от их разработчика), но конфигурирование файлов напрямую более эффективно, особенно при удаленной настройке, хотя, на мой взгляд, при этом выше вероятность совершить ошибку.

2.11. Обновление

Нет, мы сейчас не будем говорить о том, как обновлять ОС Linux, — мы только что установили ее. Я лишь хочу сказать, что обновление ОС и ее сервисов является необходимым и самым важным в стратегии построения безопасной среды. Очень сложно создать абсолютно идеальную и безопасную систему — почти везде есть ошибки, потому что программы пишет человек.

Программисты, которые пишут код, — это тоже люди, которым свойственно ошибаться. Они ошибаются, и когда кто-то находит эти ошибки (сами программисты или хакеры), выпускаются патчи/исправления, которые необходимо устанавливать, и желательно это делать как можно скорее после их появления.

Уязвимости иногда появляются раньше, чем вы успеваете скачать и установить программу. Регулярно следите за появляющимися угрозами и обновляйте свою систему, и ваш сон будет более спокойным.

2.12. Устройство Linux: ядро и модули

Я не буду здесь вдаваться в самые глубокие дебри устройства Linux, но о базовых вещах мы поговорим, и я постараюсь затронуть лишь те вопросы, которые могут вам пригодиться.

Сердце любой ОС — это ее ядро (Kernel). У Linux ядро расположено в каталоге `/boot` и имеет имя файла `vmlinuz-xxxx`, где `xxxx` — номер версии. Версии именно ядра, а не ОС. Если на последней Ubuntu 20.04 выполнить команду:

```
ls /boot
```

то она покажет содержимое этой папки, и среди ее файлов вы должны увидеть файл `vmlinuz-5.4.0-21-generic` (как я уже отмечал ранее, новый дистрибутив включает в себя ядро 5.4). Возможно, к тому моменту, как вы читаете эту книгу, вышло уже новое

2.13. Установка дополнительных пакетов в Ubuntu

Давным-давно, когда только-только развалился Советский Союз и сахар давали по талонам, программы в Linux устанавливались с помощью их компилирования из исходников. Это давало возможность ОС работать на разном «железе», а UNIX-системы, к которым относится и Linux, тогда приходилось ставить не только на процессоры от Intel, но и на процессоры IBM или Sun, которые не были совместимы с инструкциями Intel. В те времена имело смысл распространять программы в исходных кодах на языке C и из этого исходного кода собирать программы под разные платформы.

И даже компилируя систему под процессор от Intel, можно добиться максимальной производительности, если компилировать ее под определенную его версию. Два Intel-процессора разных поколений могут поддерживать разные инструкции. Если скомпилировать систему под старый процессор, то программа будет работать на большем количестве «железа». А если скомпилировать под более новый, то может работать быстрее. Так что у распространения программ в исходных кодах есть свой плюс.

Все меньше разнообразия и все больше пользователей компьютеров в мире. В такой ситуации не имеет смысла заставлять пользователей компилировать программы из исходных кодов, и разработчики Red Hat решили создать более простой метод установки — RPM. Это программа и формат файла в одном флаконе, которая устанавливает уже заранее откомпилированные приложения. Такие пакеты достаточно легко найти в Интернете или в специальных репозиториях производителей дистрибутивов.

Разработчики Debian (на нем основана Ubuntu) пошли другим путем. Они решили создать более продвинутый инструмент, который сможет не только устанавливать программу, но и следить за зависимостями. Дело в том, что для работы какой-либо определенной программы может понадобиться определенная библиотека, и нужно, чтобы программа установки могла самостоятельно находить такие зависимые библиотеки и устанавливать их в систему.

Зависимости в Linux были серьезной проблемой, особенно в начале ее пути, потому что приложения могли не работать по непонятным причинам (в основном из-за отсутствия нужной библиотеки), и все это приходилось решать «руками». Это одна из причин, по которой Linux в свое время завоевала славу сложного дистрибутива, хотя уже давно таковой не является.

Итак, Debian создает свой инструмент и называет его APT (Advanced Packaging Tool), и он до сих пор используется в дистрибутивах, ведущих свое происхождение от Debian, включая Ubuntu.

Что ж, не успели мы установить систему, как тут же начинаем говорить о ее расширении с помощью установки дополнительных программ... В Ubuntu для этого служит команда `apt-get`. Она потребует прав администратора, поэтому выполнять ее необходимо из-под `sudo`.

Если какая-то программа была удалена, но остались зависимости, то можно автоматически удалить все лишнее:

```
sudo apt-get autoremove
```

Для установки обновлений уже установленных программ выполняем следующую команду:

```
sudo apt-get upgrade
```

Чтобы обновить саму ОС, выполняем команду:

```
sudo apt-get dist-upgrade
```

Следующую тройку команд желательно выполнять регулярно для установки обновлений:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

```
sudo apt-get dist-upgrade
```

Все эти команды позволят вам содержать вашу копию релиза Ubuntu и установленных программ в обновленном состоянии, но на новый релиз автоматически вы так не перейдете. Если вышел новый релиз, а не просто обновление существующего, то на него можно перейти командой:

```
sudo do-release-upgrade
```

Чтобы узнать, какие вообще в системе уже установлены пакеты, можно выполнить следующую команду

```
sudo apt list --installed
```

2.14. Установка дополнительных пакетов в CentOS

В CentOS — другой менеджер пакетов, и хотя он работает примерно так же, различия все же есть. Чтобы установить уже упоминавшуюся в *разд. 2.13* программу Midnight Commander (команда `mc`), нужно выполнить следующую команду:

```
sudo yum install mc
```

Здесь `mc` — имя пакета, который мы хотим установить.

Обновление пакета `mc`:

```
sudo yum update mc
```

Если не указывать имя пакета, то команда `yum` обновит все установленные пакеты.

Проверить на наличие обновлений можно так:

```
sudo yum check-update
```

А удалить пакет так:

```
sudo yum remove mc
```




ГЛАВА 3

Добро пожаловать в Linux

В этой главе мы начнем знакомиться с самой Linux. Надеюсь, что вы уже установили систему, а не просто прочитали предыдущую главу, потому что все, что мы станем здесь рассматривать, желательно тут же проверять на практике. При этом материал лучше отложится в памяти и усвоится — по крайней мере, так происходит со мной. Практика — лучший учитель.

Нам предстоит поближе познакомиться с файловой системой, основными конфигурационными файлами и командами, которые пригодятся в каждодневной работе. ОС Linux может работать в двух режимах: графическом и текстовом. Я буду больше внимания уделять текстовому режиму и командной строке, потому что мы все же говорим о безопасности, которая в большей степени важна для серверов, где графический режим не используется. Конечно, графический режим можно включить и на сервере, но это там просто ненужное излишество.

Я постараюсь показать вам преимущество консоли перед курсором мыши. Дело в том, что серверы зачастую обходятся даже без монитора и, как правило, размещены где-то в отдаленной серверной. Управление ими производится через удаленную консоль, и визуальные возможности Linux не задействуются. Тогда зачем загружать тяжелые графические библиотеки, файлы и прочие ресурсы поддержки графики? На мой взгляд, это бессмысленное расходование памяти. Кроме того, при подключении к удаленной системе через SSH расход трафика невелик, и эффективно управлять сервером получается даже через мобильный Интернет.

Серверы, которые я поддерживаю, находятся в США, а я работаю из Торонто, и единственный способ получить возможность управлять этими серверами — подключаться к ним удаленно из командной строки.

Конечно, не все компьютеры на базе Linux являются серверами — домашние станции тоже могут работать на этой ОС, и тут удобнее графический интерфейс, который необходим для работы с пользовательскими утилитами и программами.

Как видите, возможность работать в двух режимах — это преимущество Linux, а не недостаток. С недавних пор в Windows также появилась возможность загружать только ядро и работать в командной строке без графической оболочки, что позво-

ляет сэкономить память и повысить надежность этой ОС. Когда не включены графические библиотеки, то и проблемы с ними отсутствуют. Сколько раз мы видели синие экраны с ошибками в драйвере видеокарты?

3.1. Файловая система

Прежде чем перейти к настройкам системы, нам нужно познакомиться поближе с файловой системой Linux. О ее структуре мы уже немного поговорили в *разд. 2.3*, когда занимались разбиением жесткого диска на разделы. В табл. 2.1 были представлены разделы, которые можно создать в Linux, а это не что иное, как основные папки, находящиеся в корне файловой системы.

Теперь, наверное, нужно было бы рассмотреть команды, с помощью которых можно управлять каталогами и файлами, а также просматривать и редактировать их. Но мы займемся этим чуть позже, а сейчас я хочу рассказать вам об одной очень полезной программе — Midnight Commander (MC). Это лучшее средство для решения всех описанных ранее задач. Программа присутствует в большинстве дистрибутивов. Для ее запуска наберите в командной строке `mc` и нажмите клавишу <Enter>. Если программа на компьютере не обнаружится, то ее надо установить, и как это сделать, было описано в *разд. 2.13*.

Файловый менеджер Midnight Commander является аналогом знаменитого файлового менеджера Norton Commander, работавшего в MS-DOS. Их функции и даже внешний вид очень похожи. И если в MS-DOS такая утилита была необходимостью из-за слабых возможностей командной строки, то в Linux это приятное и удобное дополнение. Так что, если вы знакомы с той памятной утилитой, вам будет очень приятно ностальгировать по старым добрым временам MS-DOS, сидя в современной Linux. Программа может работать как в текстовом режиме Linux, так и в окне терминала оконного менеджера.

Почему я начинаю знакомство с файловой системой с этой программы? Она проста, удобна и наглядна. Не нужно запоминать множество команд командной строки, и с этой утилитой любят работать не только начинающие, но и опытные профессионалы.

На рис. 3.1 показано окно терминала, в котором запущена программа MC. Окно состоит из двух независимых панелей, в каждой из которых вы можете видеть файлы и папки текущего каталога (имена папок начинаются со слэша — прямой наклонной черты /). Для перемещения между панелями служит клавиша <Tab>.

На правой панели открыта корневая папка — это самый верхний уровень файловой системы. Посмотрите на представленный там список папок — большинство названий нам знакомо из табл. 2.1. Каждая из этих папок может находиться в собственном разделе жесткого диска, если при установке вы его создали. Но в любом случае в файловой системе вы будете видеть их все единым блоком в составе корневой папки.

В *разд. 2.3.2* мы говорили про корневой каталог, который в Linux обозначается так: / . Именно он является вершиной пирамиды в иерархии всех папок. Пусть папки

- ❑ **6 RenMov** (Переместить) — переместить выделенные файлы или папки. По умолчанию файл будет перенесен в каталог, являющийся текущим для противоположной панели программы MC;
- ❑ **7 Mkdir** (Новый каталог) — создать новый каталог в текущем;
- ❑ **8 Delete** (Удалить) — удалить выделенные файлы и папки;
- ❑ **9 PullDn** (Выпадающее меню) — вызвать меню программы MC, которое находится в верхней части окна;
- ❑ **10 Quit** (Выход) — выход из программы.

Файлы и папки, имена которых начинаются с точки, чаще всего являются конфигурационными, и по идее они должны быть скрыты. Будьте осторожны при их перемещении и редактировании — конфигурационные файлы нуждаются в максимальной защите. Конечно, вы можете создать собственный файл с именем, у которого в самом начале стоит точка и который не будет конфигурационным. Есть также множество файлов без начальных точек в имени, являющихся конфигурационными. Так что точка — это не обязательный индикатор, но к вниманию призывает.

На начальном этапе, когда нужно исследовать систему и много бродить по каталогам и просматривать файлы, программа MC может быть очень полезной. Впоследствии, когда вы ближе познакомитесь со структурой файловой системы, вам намного проще будет выполнить команду в консоли.

3.1.1. Основные команды

Давайте рассмотрим основные команды файловой системы, которые мы будем использовать в книге, и заодно подробнее познакомимся с файловой системой Linux. Эти команды выполняются уже не в программе MC, а прямо в консоли.

pwd

Эта команда выводит на экран полный путь к текущему каталогу. С ее помощью вы можете в любой момент узнать, где находитесь, если вдруг заблудитесь в этом прекрасном пингвиньем лесу.

ls

Команда `ls` выводит список файлов и подкаталогов указанного каталога. Если имя каталога (файла) отсутствует в параметрах команды, то отображается содержимое текущего каталога. По умолчанию все настроечные файлы (имена которых начинаются с точки) являются скрытыми. Чтобы их вывести, нужно указать ключ `-a`:

```
ls -a
```

Если мы, кроме этого, хотим увидеть не только имена (сжатый формат), но и полную информацию о файлах в каталоге, нужно добавить ключ `-l`. То есть выполнить команду:

```
ls -al
```


Это работает, если файл находится в текущем каталоге. А если нет? Тогда надо указать полный путь:

```
cat /home/root/need.txt
```

tac

Эта команда, обратная команде `cat` (даже название команды — это слово `cat` наоборот), и она выводит на экран файл в обратном порядке, начиная с последней строки до первой.

cd

Команда `cd` позволяет сменить текущий каталог. Для этого необходимо в качестве параметра задать нужную папку:

```
cd /home/flenov
```

Если вы находитесь в каталоге `/home` и хотите перейти в папку `flenov`, которая располагается внутри текущей, то достаточно набрать только имя папки `flenov`:

```
cd flenov
```

Если нужно переместиться на уровень выше — например, из подкаталога `/home/flenov` в каталог `/home`, то выполняем команду:

```
cd ..
```

Как мы уже знаем, папка с именем из двух точек указывает на родительский каталог. Если перейти в нее, то мы попадем на уровень выше в структуре файловой системы.

Символ `~` указывает на домашний каталог. Если нужно быстро переместиться в него, то выполняем команду:

```
cd ~
```

Этот же символ можно использовать и в тех случаях, когда нужно просто обратиться к файлу в своем домашнем каталоге. Например, путь `~/readme.txt` приведет к файлу `readme.txt` в домашней папке текущего пользователя.

cp

Это команда копирования файла. С ее помощью можно выполнять несколько различных действий:

- ☐ копировать содержимое файла в другой документ той же папки:

```
cp /home/root/need.txt /home/root/need22.txt
```

Здесь содержимое файла `/home/root/need.txt` (источник) будет скопировано в файл `/home/root/need22.txt` (назначение);

- ☐ копировать файл в другой каталог:

```
cp /home/root/need.txt /home/flenov/need.txt
```


После ключа `-name` можно указать имя файла или шаблон, по которому нужно искать файлы. Например, следующая команда найдет на диске все файлы с именем `passwd`:

```
find / -name passwd
```

А что, если вы не помните, какое точно имя у файла: `passwd` или `password`? В этом случае можно указать следующий шаблон:

```
find / -name passw*d
```

Под этот шаблон попадают оба имени файла. Символ звездочки соответствует любому набору любых символов и даже отсутствию символов.

Ключ `-name`, наверное, самый популярный ключ для команды поиска, но помимо него команда поддерживает так же следующие ключи:

- ❑ `-size` *число* — позволяет указать размер файла, если вы его точно знаете. По умолчанию размер указывается в блоках размером в 512 Кбайт. Если нужно указать размер в байтах, то после числа укажите букву `c`;
- ❑ `-print` — заставляет вывести содержимое файла в консоль;
- ❑ `-mtime` *число* — в качестве параметра *число* можно указать количество дней, прошедших со дня последнего редактирования файла. Например, если вы вчера редактировали файл с именем `note`, но не помните, куда его сохранили, то выполните следующую команду:

```
find / -name note -mtime 1
```

- ❑ `-type` *тип* — ключ позволяет указать тип файла. В качестве типа можно указывать:
 - `d` — каталог;
 - `f` — обычный файл;
 - `p` — именованный канал;
 - `l` — символическую ссылку (о ссылках — символических и жестких — рассказано в *разд. 3.1.3*).

С помощью команды `find` можно искать не только имена файлов или каталогов, но и осуществлять поиск по содержимому файлов. Вот вам интересная команда, которую нужно запомнить или записать:

```
find . -type f -name "*" -print | xargs grep "текст, который ищем"
```

В этом примере в качестве начального каталога для поиска указана точка, т. е. поиск будет начат с текущей папки, но вы можете указать и другой путь поиска, потому что это просто пример. Потом указываем тип искомых файлов — при этом ограничиваемся только файлами, чтобы программа не пыталась мучить каталоги. В качестве имени указываем только звездочку, что соответствует всем файлам. Ключ `-print` заставляет отображать содержимое файла.

Далее идет самое интересное. После вертикальной черты стоит другая команда. Это значит, что результат работы команды слева от черты будет передаваться команде

rm

Команда позволяет удалить файл или каталог (который должен быть пуст):

```
rm /home/flenov/need22.txt
```

В качестве имен файлов можно использовать и маски, как в команде `cp`. Для удаления каталога может понадобиться указание следующих ключей:

- ☐ `-d` — удалить каталог;
- ☐ `-r` — рекурсивно удалять содержимое каталогов;
- ☐ `-f` — не запрашивать подтверждение на удаление файлов. Будьте внимательны при использовании этого ключа, потому что файлы будут удаляться без каких бы то ни было предупреждений. Вы должны быть уверены, что команда написана правильно, иначе можно удалить что-то нужное, особенно если вы работаете под учетной записью `root`.

Пример удаления каталога рекурсивно и без запроса на подтверждение:

```
rm -rf /home/flenov/dir
```

df

Эта команда позволяет узнать объем свободного места на жестком диске или в разделе. Если устройство не указано, то на экран выводится информация о смонтированных файловых системах.

В результате выполнения команды может быть получен примерно такой вывод:

Filesystem	1k-blocks	Used	Available	Use%	Mounted on
/dev/hda2	16002200	2275552	12913760	15%	/
none	127940	0	127940	0%	/dev/shm

Результирующая таблица состоит из следующих колонок:

- ☐ `Filesystem` — диск, файловая система которого смонтирована;
- ☐ `1k-blocks` — количество логических блоков;
- ☐ `Used` — количество использованных блоков;
- ☐ `Available` — количество доступных блоков;
- ☐ `Use%` — процент использованного дискового пространства;
- ☐ `Mounted on` — как смонтирована файловая система.

mount

Команда предназначена для монтирования файловых систем. Были времена, когда даже компакт-диск (CD-ROM) приходилось монтировать командой в терминале. Сейчас в графическом режиме все уже монтируется и демонтируется автоматически. Да и нужно ли в наши дни описывать эту команду или пора забыть о таких носителях?

Если вы создали какие-то разделы на отдельных дисках, то с помощью файла `mtab` сможете настраивать и их. Ранее я вам рекомендовал выделить таким образом раздел `/home` с пользовательскими каталогами. Если вы так и сделали, то в файле `mtab` может присутствовать еще одна строка примерно следующего вида:

```
/dev/hda3 /home ext3 rw,errors=remount-ro 0 0
```

Посмотрим на ее четвертый параметр. В нем через запятую указаны опции монтирования, которыми можно управлять для повышения безопасности системы. В нашем примере это: `rw,errors=remount-ro`. Они означают, что по умолчанию диск будет смонтирован для чтения и записи (`rw`), а при возникновении ошибок перемонтирован только для чтения (`errors=remount-ro`). В качестве опций монтирования дополнительно можно указывать:

- ❑ `noexec` — запрещает выполнение файлов. Если вы уверены, что в разделе не должно быть исполняемых файлов, то можно указать эту опцию. Например, в некоторых системах каталог `/home` должен хранить только документы. Чтобы хакер не смог записать в этот раздел свои программы, с помощью которых будет осуществляться взлом, добавьте эту опцию. Точнее сказать, программы записать он сможет, а запустить — нет;
- ❑ `nosuid` — запрещает использование программ с битами SUID и SGID. В разделе `/home` их быть не должно, поэтому можно явно запретить применение привилегированных программ. О SUID- и SGID-программах мы будем говорить еще не раз (например, в *разд. 4.5*), ибо это очень опасная вещь;
- ❑ `nodedv` — запрещает использование файлов устройств;
- ❑ `nosymfollow` — запрещает использование мягких ссылок.

Опции `nodedv` и `nosymfollow` не сильно влияют на безопасность, но могут пригодиться.

Вообще-то использование упомянутого ранее параметра `noexec` — бесполезное занятие и абсолютно не защищает систему от профессионального хакера, потому что опытный взломщик сможет запустить программу, если для выполнения разрешен хотя бы один раздел. Нет, параметр использовать можно, но не стоит считать его панацеей.

Допустим, что ваш сайт разрабатывается с использованием языка Perl. Если его интерпретатор доступен для выполнения, то взломщик сможет запускать сценарии программ Perl в любом разделе, в том числе и с установленным параметром `noexec`. Так, при использовании для запуска сценария командной строки вы получите сообщение о нарушении прав доступа. Но программа выполнится, если отдать такую команду:

```
perl file.pl
```

В этом случае, несмотря на то что файл `file.pl` находится в разделе, в котором запрещены исполняемые файлы, ошибки не будет, потому что запускается разрешенная программа `perl`, которая, в свою очередь, читает файл (разрешенная операция)

tar

В те времена, когда не было программ для установки пакетов — таких как `apt-get`, программы устанавливались из архивов, распаковываемых с помощью команды `tar`. Сейчас таким образом чаще всего распространяют программы в исходных кодах. Для развертывания файла архива `tar` нужно выполнить команду:

```
tar xzvf имяфайла.tar.gz
```

Как правило, после выполнения команды в текущем каталоге будет создан каталог с таким же именем, как у архива (только без расширения). В нем вы сможете найти все распакованные файлы.

К работе с архивами мы вернемся в *главе 13*, когда станем рассматривать резервирование данных. Сейчас же нам достаточно уметь распаковывать пакеты, чтобы устанавливать дополнительные программы и утилиты сторонних разработчиков.

rpm

Эта команда устанавливает пакеты программ формата RPM, и ее смысл такой же, как и у `apt-get`, только формат RPM был разработан в Red Hat и используется в дистрибутивах, построенных на его базе. Я же рассматриваю в этой книге Ubuntu — потомок Debian. Но все же проведем совсем уж краткий экскурс по команде `rpm`.

Для установки нового пакета можно выполнить команду:

```
rpm -i пакет
```

Для обновления уже установленного пакета можно выполнить команду с параметром `-U`:

```
rpm -U пакет
```

Для того чтобы видеть ход инсталляции, можно указать еще и ключ `-v`. Таким образом, команда установки будет выглядеть следующим образом:

```
rpm -iv пакет
```

which

Иногда необходимо знать каталог, в котором расположена программа. Для этого служит команда `which` с именем программы в качестве параметра, которая проверит основные каталоги, содержащие исполняемые файлы. Например, чтобы определить, где находится программа просмотра содержимого каталогов `ls`, выполните следующую команду:

```
which ls
```

В результате вы увидите путь: `/bin/ls`. Если ваша ОС поддерживает псевдонимы (`alias`) команд, то можно будет увидеть и его. Таким образом, при выполнении команды на экране выведется:

```
alias ls='ls -color=tty'
/bin/ls
```


Рассмотрим по частям эту строку:

- ❑ `find` — программа поиска файлов;
- ❑ `каталог` — каталог, в котором нужно искать. В нашем случае указан системный каталог `/etc`, в котором хранятся все настроечные файлы;
- ❑ `параметр \ ( -сравнение файл \)` состоит из файла для сопоставления и критерия поиска файлов, который может принимать различные значения:
 - `-newer` — дата изменения позже, чем у заданного файла в параметре файл;
 - `-cnewer` — состояние изменено позже, чем у сопоставляемого файла в параметре файл;
 - `-anewer` — дата последнего доступа превосходит аналогичный параметр сравниваемого файла;
- ❑ `параметр -ls` — отображает на экране список файлов (как при выполнении команды `ls`).

Контрольные суммы

На даты изменения можно надеяться, но необходимо иметь дополнительное средство проверки. Наилучшим методом является подсчет контрольной суммы. Допустим, что вы хотите отслеживать изменения в каталоге `/etc`. Для этого выполните следующую команду:

```
md5sum /etc/*
```

Утилита `md5sum` подсчитывает контрольную сумму указанных в качестве параметра файлов. На экране вы получите результат выполнения команды примерно такого вида:

```
783fd8fc5250c439914e88d490090ae1 /etc/DIR_COLORS
e2eb98e82a51806fe310bffdd23ca851 /etc/Mutttrc
e1043de2310c8dd266eb0ce007ac9088 /etc/a2ps-site.cfg
4543eebd0f473107e6e99ca3fc7b8d47 /etc/a2ps.cfg
c09badb77749eecbeafd8cb21c562bd6 /etc/adjtime
70aba16e0d529c3db01a20207fd66b1f /etc/aliases
c3e3a40097daed5c27144f53f37de38e /etc/aliases.db
3e5bb9f9e8616bd8a5a4d7247f4d858e /etc/anacrontab
fe4aad090adcd03bf686103687d69f64 /etc/aspldr.conf
...
```

Результат отображается в две колонки: первая содержит контрольную сумму, а вторая — имя файла. Контрольные суммы подсчитываются только для файлов. Для каталогов будет выведено сообщение об ошибке.

В нашем случае показаны все файлы каталога `/etc/*`. Результат расчета выведен на экран, но запомнить столько данных невозможно, поэтому логично будет записать их в файл, чтобы потом использовать его содержимое для анализа измене-

дет проблематично. Системные библиотеки также изменяются с каждой установкой обновлений. А вот конфигурационные файлы и программы в настроенной системе не изменяются и не должны изменяться, поэтому любые их корректировки указывают на возможную проблему или вторжение.

Замечания по работе с файлами

ОС Linux достаточно демократично относится к именам создаваемых файлов, позволяя использовать абсолютно любые символы, кроме знака /, который является разделителем каталогов, и символа с кодом 0, который определяет конец имени файла. Все остальное применять допускается.

Самое страшное — это возможность использовать невидимые символы. Дело в том, что на их основе можно создать программу, у которой в имени будут только нечитаемые знаки, и пользователь не увидит такого файла. Никто и никогда не вглядывается в имена файлов, да и администраторы не сидят в постоянном наблюдении за системой, поэтому подобные вещи легко пропустить.

Рассмотрим пример с использованием перевода строки. Допустим, что хакер назвал свой файл `hacker\nhosts.allow`. В этом случае под `\n` подразумевается перевод каретки, а, значит, имя этого файла состоит из двух строк:

```
hacker  
hosts.allow
```

Не все программы могут обработать такое имя правильно. Если ваш файловый менеджер работает неверно, то он отобразит только вторую строку: `hosts.allow`.

Еще один способ спрятать файл — в качестве имени указать точку и пробел: «. » или две точки и пробел: «.. ». Файл с именем в виде точки всегда указывает на текущий каталог. Администратор, выполнив команду `ls`, может не заметить, что существуют два файла с одинаковыми именами, а пробела ему все равно не видно.

Пробелы можно вставлять в любые имена файлов — например, перед именем: (« `hosts.allow`») или, наоборот, в конце имени, и при отсутствии должного внимания вы ничего не заметите. Чтобы увидеть конечный пробел, можно при выводе добавлять к каждому имени символ косой черты (/). Для этого при вызове команды `ls` используйте ключ `-F`.

Следующий вариант спрятать файл — заменять одни символы на другие, схожие по начертанию. Например, посмотрим на имя файла `hosts.a1low`. Ничего не замечаете подозрительного? При беглом взгляде проблему можно пропустить, но если приглядеться повнимательнее, то станет видно, что вместо букв `l` (`L`) стоит цифра `1` (единица).

Может использоваться и такой прием — подмена буквы `b` на `d`. Здесь тоже трудно что-либо заподозрить, потому что если человек каждый день видит одно и то же, то чаще всего воспринимает желаемый текст за действительный. Получается банальный обман зрения.

Подобные трюки очень часто используются вирусописателями на платформе Windows. Там опытные пользователи, которые работают с компьютерами с 1990-х го-

В ответ на это программа создаст жесткую ссылку с именем `имя_ссылки`, которая будет указывать на те же данные, что и файл `имя_файла`.

Чтобы на практике проверить все, что утверждается далее, создайте в своей системе файл `1.txt`. Для этого можно выполнить команду:

```
cat > 1.txt
```

Нажмите клавишу `<Enter>`, введите несколько строк текста и нажмите комбинацию клавиш `<Ctrl>+<D>` — теперь у вас есть необходимый файл для тестирования.

Создайте для файла `1.txt` жесткую ссылку. Для этого выполните следующую команду:

```
ln 1.txt link.txt
```

С помощью команды `cat link.txt` выведите на экран содержимое файла `link.txt` и убедитесь, что оно идентично строкам в файле `1.txt`. Теперь выполните команду `ls -il`, чтобы просмотреть содержимое каталога. В списке файлов должны быть две строки:

```
913021 -rw-r--r--  2 root    root          0 Feb 22 12:19 1.txt
913021 -rw-r--r--  2 root    root          0 Feb 22 12:19 link.txt
```

Обратите внимание, что первая колонка, в которой находятся дескрипторы для обоих файлов, содержит одинаковые значения. В третьей колонке стоит число 2, что говорит о наличии двух ссылок на данные.

Теперь попробуем изменить содержимое любого из этих файлов. Для этого выполним следующие команды:

```
ls > link.txt
cat 1.txt
```

В первой строке мы сохраняем в файле `link.txt` результат работы команды `ls` (результат отображения содержимого каталога), а вторая — отображает документ `1.txt`. Убедитесь, что содержимое обоих файлов изменилось, но данные в них записаны одинаковые.

Давайте попробуем удалить файл `1.txt` и посмотреть на каталог и содержимое файла `link.txt`. Для этого выполните следующие команды:

```
rm 1.txt
ls -il
cat link.txt
```

Файл `1.txt` будет удачно удален. А вот содержимое жесткой ссылки `link.txt` никуда не денется. То есть данные на диске не были уничтожены, а исчезло только имя `1.txt`. Обратите внимание, что у файла `link.txt` значение счетчика ссылок в третьей колонке уменьшилось до единицы.

Символьные ссылки

Символьная ссылка указывает не на данные, а на имя файла. Это дает некоторые преимущества, но одновременно вызывает большое количество проблем. Для

Глядя на все недостатки ссылок, возникает вопрос — а нужно ли действительно использовать их? Я рекомендую делать это только в крайнем случае, когда все остальные способы решения проблемы еще хуже. И если нет другого выхода, то будьте хотя бы аккуратны и внимательны.

3.2. Загрузка системы

Во время загрузки ОС запускается множество программ, которые занимают память, уменьшая тем самым производительность системы. Желательно, чтобы при старте сервера запускались только те демоны, которые действительно нужны, и в память не загружалось ничего лишнего.

В процессе загрузки все необходимые настройки должны производиться автоматически, чтобы сразу после старта не приходилось что-либо конфигурировать вручную. Это может отнять слишком много времени, к тому же выполнять одни и те же действия каждый раз — очень скучно и неинтересно.

3.2.1. Автозагрузка

Когда включается компьютер, то первой загружается программа BIOS (Basic Input/Output System, базовая система ввода/вывода), записанная в микросхеме ПЗУ системы. Она читает главную загрузочную запись (MBR, Master Boot Record), занимающую 512 байтов на жестком диске. В них содержится информация о дисках и код начального загрузчика ОС.

Поскольку уместить код всего загрузчика в MBR в наше время практически невозможно, там располагают только начальный загрузчик, который загружает с другого места диска уже полноценный загрузчик.

В Linux сейчас самым популярным загрузчиком является GRUB2. Конфигурация этой программы чаще всего содержится в файле `/boot/grub/grub.cfg`, но можно встретить дистрибутивы, где этот путь выглядит так: `/boot/grub2/grub.cfg`. В файле загрузчика записаны настройки, определяющие, нужно ли показывать меню выбора варианта загрузки, и каким должно быть это меню, а также указываются сами варианты загрузки и предписывается, какую версию ядра следует загрузить (мы уже говорили, что в Linux может быть более одного ядра).

После этого может формироваться `initrd` — виртуальный диск в оперативной памяти (`ramdisk`) для временной файловой системы, используемой ядром Linux при начальной загрузке. Дело в том, что если ядро скомпилировано в минимальном варианте и все его возможности подключаются в дальнейшем с помощью модулей, то в этот момент у ядра не будет возможности работать со специфичными жесткими дисками, и оно не сможет загружать дальнейшие данные. Поэтому сначала грузится диск `/boot/initrd.xxxx`, который загружает критически важные модули, и, если я не ошибаюсь, именно он и загружает красивую картинку, которую показывает ОС во время старта системы.

Чтобы просмотреть состояние всех установленных служб в тот или иной момент, надо выполнить команду:

```
sudo service --status-all
```

Когда вы запускаете или останавливаете сервис с помощью подобных команд, то это носит только временный характер. Если же вы хотите, чтобы изменение носило постоянный характер и состояние службы сохранялось после перезапуска, то можно создать файл `/etc/init/имя_сервиса.override`, в котором будет содержаться только слово `manual`. Из командной строки такое можно выполнить с помощью команды `echo` и перенаправить результат команды в файл:

```
echo 'manual' > /etc/init/mysql.override
```

Команда `echo 'manual'` говорит о том, что нужно напечатать слово `manual`. В общем случае оно будет напечатано на стандартном выводе, которым по умолчанию является окно терминала. Но имеющийся в строке символ `>` говорит, что вывод команды слева нужно перенаправить в поток справа. А справа идет файл. Значит, теперь слово будет напечатано не на экране, а записано в файл.

Папка `/etc` требует прав администратора для работы с ее файлами, а это значит, что команда завершится ошибкой, если ее выполнять с правами простого пользователя.

Проблема решается использованием команды `sudo`:

```
sudo bash -c "echo 'manual' > /etc/init/mysql.override"
```

Фрагмент `bash -c` говорит о том, что нужно просто выполнить команду, которая идет дальше в кавычках. Использование команды `bash -c` вызвано тем, что команда `sudo` не сможет работать с командой `echo` и перенаправить наш вывод в файл, потому что `echo` — это команда не ОС, а `bash`.

3.2.2. GRUB2

Операционная система — когда вы включаете свой компьютер — сама по себе не загружается. Ее запуск должен быть как-то инициализирован. Изначально самым популярным инициализатором этой загрузки был загрузчик Linux под названием LILO (Linux Loader), но в последнее время уже, кажется, все дистрибутивы перешли на GRUB2 (GRand Unified Bootloader, большой объединенный загрузчик, версия 2).

GRUB2 — один из самых стабильных и развитых проектов. 2-я его версия вышла в 2012 году, а в 2019-м появилась версия 2.04, представляющая собой последнюю стабильную версию этого загрузчика, и она включена в Ubuntu 20.04. Представьте себе, что за 7 лет номер версии увеличился с 2.0 всего до 2.04. Это настолько развитый продукт, что новые изменения выходят очень редко.

Как уже отмечалось ранее, когда мы включаем компьютер, первым запускается система BIOS. Ее загрузка сейчас может быть скрыта красивой картинкой с логотипом производителя, но раньше в этот момент мы обязательно видели логотип BIOS, а по экрану бежали циферки тестирования оперативной памяти.

□ информация, которая будет отображаться на экране после входа в систему, конфигурируется через файлы из каталога `/etc/update-motd.d`. Здесь находится сразу несколько файлов скриптов, и вы можете редактировать уже существующие или создавать новые. Имена файлов имеют формат `XX_filename`, где `XX` — порядковый номер, который определяет, в каком порядке будут загружаться файлы. Потом просто следует имя. Такой же формат используется при конфигурации GRUB2 и некоторых других программ.

3.3. Регистрация в системе

В процессе старта ОС загружает виртуальные консоли `getty`. Каждая из них для работы требует авторизации и запрашивает имя пользователя. Для этого на экран выводится окно приглашения. Введенное имя пользователя передается программе `login`, а она, в свою очередь, запрашивает пароль.

Программа `login` сравнивает имя пользователя со списком имен в файле `/etc/passwd`, а пароль — с соответствующей записью в файле `/etc/shadow`. Все пароли в этом файле хранятся только в зашифрованном виде. Поэтому для сопоставления введенный пароль тоже шифруется и результат сравнивается с зашифрованным значением пароля в файле `/etc/shadow` для указанного имени пользователя.

Почему проверка происходит так сложно? Дело в том, что все пароли в файле `/etc/shadow` зашифрованы необратимым алгоритмом. Это значит, что математическими методами из результата шифрования получить исходный пароль нельзя, поэтому остается только подбор. Для подбора паролей существуют несколько очень простых программ. Чем проще пароль и чем меньше его длина, тем быстрее программа найдет нужный вариант. Когда пароль сложен и его длина более 8 символов, а лучше — свыше 16, то подбор отнимет достаточно много ресурсов.

Если идентификация пользователя состоялась, то программа `login` выполнит все автоматически загружаемые сценарии и запустит оболочку (командную строку), через которую и будет происходить ваша работа с системой. Если проверка прошла неудачно, то система вернет управление консоли `getty`, которая снова запросит ввод имени пользователя.

Таким образом, пока мы не пройдем авторизацию через программу `login`, запустить оболочку (текстовую или графическую) невозможно — нам останется доступной лишь консоль `getty`, которая умеет только запрашивать имя пользователя и передавать его программе `login`.

Теперь обсудим ряд проблем, которые могут возникнуть при входе в систему, и посмотрим, как они решаются.

3.3.1. Теневые пароли

В очень старых версиях Linux список пользователей и пароли хранились в файле `/etc/passwd` в открытом виде. Это было не очень хорошо с точки зрения безопасности, потому что файл `/etc/passwd` должен быть доступен для чтения из-за того, что


```
root:$1$1emP$XMJ3/GrkltC4c4h/:12726:0:99999:7:::  
bin:*:12726:0:99999:7:::  
daemon:*:12726:0:99999:7:::
```

Здесь также имеется несколько параметров, разделенных двоеточиями. Нас будут интересовать первые два: имя пользователя и пароль. По имени пользователя происходит связь записи из файла `/etc/shadow` с файлом `/etc/passwd`. А вот во втором параметре уже находится настоящая зашифрованная версия пароля. Но если вместо пароля в строке стоит звездочка, это запрещает соответствующему пользователю интерактивную работу с системой. Например, для пользователей `bin` и `daemon` установлены именно звездочки — значит, под этими учетными записями нельзя входить на компьютер.

В последней версии Ubuntu пароль пользователя `root` не спрашивают, и если посмотреть на файл `shadow`, то здесь будет стоять вместо пароля звездочка — указатель на то, что этот аккаунт нельзя использовать для входа в систему. Можно установить пароль, и тогда вход снова будет возможен, но не стоит этого делать. Не зря же разработчики пошли на такой шаг.

3.3.2. Забытый пароль

Что делать, если вы забыли пароль или хакер проник в систему и изменил его? Действительно, ситуация не из приятных, но все решаемо. Если у вашей учетной записи есть доступ к файлу `/etc/shadow`, то можно заняться его редактированием, а если нет, то посмотрим, как получить к нему доступ с помощью загрузочной дискеты (или флешки, что сейчас более предпочтительно).

Загрузившись с флешки, вы должны войти в систему как `root` и подключить тот жесткий диск (или раздел диска), на котором находится папка `/etc`. Для этого нужно выполнить команду:

```
/sbin/mount -w hda1 /mnt/restore
```

Теперь каталог `/mnt/restore` (желательно, чтобы он существовал до выполнения команды) указывает на главный раздел вашего жесткого диска, а файл паролей находится в каталоге `/mnt/restore/etc/shadow`. Откройте этот файл в любом редакторе и удалите пароль администратора `root` (просто сотрите весь текст между первым и вторым знаками двоеточия). В моем случае получилось так:

```
root::12726:0:99999:7:::
```

Теперь обычным способом загружайте систему и в качестве имени пользователя вводите имя `root`. У вас даже не спросят пароль, потому что он «пустой». Не забудьте потом поменять пароль, иначе ситуация останется «опасной для жизни» — это как электрику работать под высоким напряжением без средств защиты ☺.

Для смены пароля вы должны набрать команду `passwd root`, в ответ запустится программа, которая попросит вас дважды ввести символы пароля. Такой подход исключает случайную опечатку при наборе и может с большой долей вероятности гарантировать, что будет сохранен верный пароль. Ведь если вы случайно наберете не тот символ и не заметите этого, то не сможете потом воспроизвести установлен-

Рассмотрим пример конфигурационного файла для FTP-сервиса, который находится в файле `/etc/pam.d/ftp`. Символом `↵` здесь обозначен перенос длинной строки:

```
#%PAM-1.0
auth      required /lib/security/pam_listfile.so item=user sense=deny ↵
file=/etc/ftpusers onerr=succeed
auth      required /lib/security/pam_stack.so service=system-auth
auth      required /lib/security/pam_shells.so
account   required /lib/security/pam_stack.so service=system-auth
session   required /lib/security/pam_stack.so service=system-auth
```

Это и все, что вам необходимо знать. Остальное берет на себя программа сервиса без нашего вмешательства.

3.3.4. Сложность паролей

Теперь перейдем к конфигурированию сложности самого пароля. Эта конфигурация находится в файле `/etc/pam.d/common-password`. В некоторых системах конфигурацию разбивают на несколько файлов — в зависимости от того, каким способом авторизуется пользователь, и вы можете увидеть в каталоге `/etc/pam.d/` сразу несколько файлов, имена которых имеют формат `nnnn-auth`, где на месте `nnnn` может быть одно из следующих слов: `syste`, `smartcard`, `fingerprint`. То есть в зависимости от того, каким способом авторизуется пользователь, подключается тот или иной файл конфигурации.

У нас подопытная система Ubuntu, и у нее параметры конфигурирования сложности пароля находятся в файле `/etc/pam.d/common-password`. Откройте этот файл в текстовом редакторе и добавьте следующую строку:

```
password requisite pam_cracklib.so try_first_pass retry=3 type= ucredit=-2
lcredit=-4 dcredit=-3 ocredit=-1
```

Теперь разберемся, что тут написано. Для управления паролями нужно загрузить модуль `pam_cracklib.so`. В качестве параметров указывается:

- ❑ `ucredit` — сколько букв в верхнем регистре должен содержать пароль. В моем примере прописано 2, а, значит, в пароле как минимум должны быть две буквы в верхнем регистре. Чтобы проще запомнить — первая буква `u` явно указывает на слово `upper` (верх);
- ❑ `lcredit` — сколько строчных букв должен содержать пароль. В моем примере прописано 3. Тут для простоты снова смотрим на первую букву `l` — она явно указывает на слово `lower` (низ);
- ❑ `dcredit` — сколько цифр должен содержать пароль. Первая буква `d` явно происходит от слова `digit`;
- ❑ `ocredit` — сколько других символов (не букв и не цифр) должен содержать пароль. Я тут точно не уверен, но мне кажется, что первая буква может указывать на слово `other`.

В ответ на это вы не увидите файла помощи, а на экране появится только строка:

[1] 2802

После этого терминал снова готов работать, потому что центральный процесс запустил команду `man ls` в фоновом режиме и свободен для выполнения новых директив.

А что же мы увидели в ответ на выполнение команды? В квадратных скобках показан порядковый номер фонового процесса, который мы запустили. Номера будут последовательно увеличиваться. В нашем случае это первая команда, поэтому в квадратных скобках стоит единица. Номер фонового процесса формируется для каждого пользователя. Если войти в систему через второй терминал и запустить какой-либо фоновый процесс, то вы увидите примерно следующее:

[1] 2805

В квадратных скобках опять число 1, а вот следующее значение отличается от введенного на первом терминале. Это PID (Process ID, идентификатор процесса) созданного процесса, который является уникальным для всех пользователей. Это значит, что если вы запустили процесс с номером 2802, то другой пользователь никогда не сможет запустить процесс с таким же идентификатором, по крайней мере до рестарта системы. Его PID будет другим.

Запомните идентификаторы, которые вы увидели, впоследствии они нам пригодятся.

Чтобы узнать, какие процессы у вас запущены, выполните команду `jobs`. В ответ на это вы получите:

[1] + Stopped man ls

Здесь мы видим, что процесс с номером [1] загружен в память, и состояние команды `man ls` — **Stopped** (остановлен).

Какой смысл в том, что мы отправили просмотр файла помощи в фоновый режим? Я не зря выбрал эту команду, потому что в этом есть резон. Вы в любой момент можете сделать фоновый режим основным. Для этого необходимо ввести команду `fg %1`, где число 1 указывает номер вашего процесса, который вы видели в квадратных скобках. Попробуйте сейчас выполнить эту команду, и перед вами откроется запущенная программа `man`, отображающая файл помощи по использованию команды `ls`.

Раз процесс можно сделать центральным, значит, можно поступить и наоборот. Чтобы вернуть процесс в фоновый режим, нажмите комбинацию клавиш `<Ctrl>+<Z>` — перед вами снова появится командная строка. Выполните команду `jobs`, чтобы убедиться, что команда `man ls` все еще работает.

Если в программе есть возможность выполнять системные команды, то вместо сочетания клавиш `<Ctrl>+<Z>` можно выполнить команду `bg %1`. Число 1 — это снова номер процесса.

Перед нами четыре колонки, которые показывают идентификатор процесса, терминал, на котором запущена программа, время работы и выполняемую команду.

Это далеко не полный список. Чтобы увидеть все процессы, необходимо выполнить команду `ps` с ключом `-a`. Но и это еще не будет весь перечень, потому что отобразятся только программы своего терминала. Если требуется полный список процессов, запущенных со всех терминалов, то нужно добавить ключ `-x`. Помимо этого, вы можете пожелать увидеть имя пользователя, под которым работает процесс, для этого добавьте ключ `-u`. В итоге исчерпывающую информацию можно получить, выполнив команду:

```
ps -aux
```

Результат работы будет таков:

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.1	1376	452	?	S	14:25	0:05	init
root	2	0.0	0.0	0	0	?	SW	14:25	0:00	[keventd]
root	3	0.0	0.0	0	0	?	SW	14:25	0:00	[kapmd]
root	5	0.0	0.0	0	0	?	SW	14:25	0:00	[kswapd]
root	6	0.0	0.0	0	0	?	SW	14:25	0:00	[bdflush]
root	7	0.0	0.0	0	0	?	SW	14:25	0:00	[kupdated]
root	530	0.0	0.1	1372	436	?	S	14:25	0:00	klogd -x
rpc	550	0.0	0.2	1516	540	?	S	14:25	0:00	portmap

Колонка `STAT` показывает состояние процесса. Здесь можно встретить следующие коды:

- ☐ `s` (Sleeping) — спящий, это нормальное состояние для сервисов, которые просыпаются только на редкие запросы клиентов;
- ☐ `R` (Runnable) — исполняемый в текущий момент;
- ☐ `T` (Traced or Stopped) — в состоянии отладки или остановлен;
- ☐ `Z` (Zombied) — зависший. Такие процессы можно смело «убивать»;
- ☐ `W` — не имеет резидентных страниц;
- ☐ `<` — обладает высоким приоритетом;
- ☐ `N` — имеет низкий приоритет.

Это основные состояния, которые вы можете увидеть у процессов в своей системе.

Если в колонке `TTY` стоит вопросительный знак, то это означает, что процесс запущен еще на этапе загрузки системы и не принадлежит какому-либо терминалу.

Здесь показан всего лишь небольшой фрагмент файла. В реально работающей системе процессов очень много, и даже при минимальном количестве запущенных сервисов может не хватить одного экрана для отображения их всех. Я люблю сохранять результат работы в текстовый файл, а потом спокойно изучать его в любом редакторе. Для этого я выполняю команду:

```
ps -aux >> ps.txt
```


В верхней строке окна (см. рис. 3.2) выводится количество пользователей, общая загрузка системы и статистика процессов: общее количество, спящие, выполняемые, зависшие и остановленные.

Помимо этого, там можно увидеть краткий отчет по использованию памяти: количество занятой и свободной оперативной памяти и размер раздела подкачки. В моем случае виртуальной машине выделен гигабайт памяти, и из них свободно чуть более 300 мегабайт, а раздел подкачки пока не используется (см. первую и вторую колонки строки KiB Mem). Почему так мало свободной памяти — ведь часто можно услышать, что Linux потребляет очень мало ресурсов? Дело в том, что реально использовано только 113 460 байтов (примерно 100 мегабайт — это третья колонка). Все остальное — кэшированные данные для повышения производительности системы (четвертая колонка).

Вторая строка цифр (KiB Swap) — это файл подкачки. Чем меньше компьютер использует swap-раздел/файл, тем быстрее он работает. Конечно, пока что этот раздел практически не используется, но если перейти в графический режим и запустить пару мощных приложений, то и swap-раздела не хватит. Но для этого виртуального сервера я даже не устанавливал графической оболочки, поэтому выделенного гигабайта хватает.

Программа `top` будет обновлять информацию о загрузке процессора с определенным интервалом времени. Для выхода из программы нажмите комбинацию клавиш `<Ctrl>+<C>`.

3.4.4. «Зомби»: поиск и устранение

В системе иногда появляются «умершие» процессы — «зомби». Я часто применяю Linux для запуска различных задач, по расписанию используя команду `cron` или просто скрипты, которые бесконечно работают в цикле и выполняют задачи в определенное время. ОС Linux в этом отношении удобна и достаточно надежна, но далеко не всегда надежны ее приложения.

Из личного опыта — очень часто возникают проблемы с программой `rsync`. Она синхронизирует файлы между двумя хранилищами, которые могут находиться на разных компьютерах. С помощью `rsync` достаточно просто реализовать синхронизацию кода на серверах для веб-серверов. Это одна из задач, которую регулярно для меня решает Linux. Однако `rsync` постоянно виснет. Решение — запускать его в виде демона. Став демоном, он работает отлично и пока у меня ни разу не зависал, но это уже отдельная тема.

Но вернемся к «зомби». Итак, у меня скрипты работают в постоянном режиме и создают новые процессы (дочерние), которые выполняют какие-то работы. Иногда бывает так, что дочерний процесс завершил свою работу, а родительский не видит этого и считает, что дочерний все еще работает. В системе образуются как бы «зомби» — которые не видят, что «тело уже умерло», и продолжают «дышать» и использовать ресурсы.

А что, если необходимо выполнять какие-то простые, но трудоемкие задачи после окончания трудового дня? Неужели придется оставаться на работе на всю ночь? Конечно же, нет. Компьютер может сам все сделать без вмешательства человека, главное — правильно ему рассказать, что и когда надо выполнить.

3.5.1. Формирование задания

Самый простой, надежный и любимый хакерами способ решить проблему запуска в определенное время — это команда `at`. В простейшем случае формат ее вызова следующий:

```
at hh:mm dd.mm.yy
```

Дату можно и не задавать, и тогда будет взята ближайшая возможная. Если заданное время раньше текущего, то будет установлена текущая дата, если позже — то следующий день, потому что сегодня эта команда уже выполниться не сможет.

Рассмотрим использование команды `at` на реальном примере, с которым вы можете столкнуться в жизни. Допустим, что в начале рабочего дня мы завели нового пользователя. Мы также знаем, что этот сотрудник будет работать до половины первого. Если после этого времени забыть удалить учетную запись, то пользователь останется в системе, а это большая дыра в безопасности.

Для начала необходимо выполнить команду `at`, указав время 12:50. Я беру запас 20 минут на случай, если пользователь задержится в системе. Для этого выполните команду:

```
at 12:50
```

В ответ на это появится приглашение для ввода команд:

```
at>
```

Введите необходимые инструкции. Например, для удаления пользователя необходимо выполнить команду `userdel` и стереть соответствующий каталог:

```
userdel tempuser  
rm -fr /home/tempuser
```

Подробнее об управлении пользователями мы узнаем в *главе 4*, а сейчас нужно довериться мне и просто использовать эти команды. Конечно же, в вашей системе сейчас нет пользователя `tempuser`, и команда не отработает, но нас интересует сам факт ее запуска в указанное время.

Наберите эти команды, в конце каждой из них нажимая клавишу <Enter>. Для завершения ввода используйте комбинацию клавиш <Ctrl>+<D>. В ответ на это вы должны увидеть текстовое сообщение, содержащее дату и время, в которое будет выполнена команда, а также идентификатор задания. Сообщение будет выглядеть примерно следующим образом:

```
Job 1 at 2005-03-03 12:50
```


Здесь заполнены только часы и минуты. Так как остальные параметры не указаны, то команда будет выполняться ежедневно в 5 утра, вне зависимости от дня недели, числа или месяца.

```
00 20 * * 1 /home/flenov/backup2_script
```

Эта команда выполняет файл сценария каждый понедельник (день недели равен 1) в 20:00.

```
00 * * * * /home/flenov/backup3_script
```

Такая команда будет выполняться каждый час ровно в 00 минут.

ВНИМАНИЕ!

Если вы запустите программу `crontab` и, не введя команд, нажмете комбинацию клавиш `<Ctrl>+<D>`, то все предыдущие задания сотрутся. Будьте внимательны — для выхода из программы без сохранения используйте только комбинацию `<Ctrl>+<C>`.

Помимо этого, у сервиса `cron` есть несколько дополнительных каталогов, упрощающих создание расписаний. Можно распределить выполняемые сценарии по каталогам:

- ☐ `/etc/cron.hourly` — ежечасно;
- ☐ `/etc/cron.daily` — ежедневно;
- ☐ `/etc/cron.weekly` — еженедельно;
- ☐ `/etc/cron.monthly` — ежемесячно.

Вроде все просто, но если сценарий выполняется еженедельно, то в какое время и в какой день недели? Все станет ясно, когда вы посмотрите на `/etc/crontab` — конфигурационный файл сервиса `cron`. В нем есть следующие строки:

```
01 * * * * root run-parts /etc/cron.hourly
02 4 * * * root run-parts /etc/cron.daily
22 4 * * 0 root run-parts /etc/cron.weekly
42 4 1 * * root run-parts /etc/cron.monthly
```

В начале строки указывается время выполнения. Вы можете изменить его так, что сценарии из каталога `/etc/cron.monthly` начнут выполняться ежечасно. Так что названия чисто символические и только по умолчанию. Они легко меняются.

Обратите также внимание, что в этих каталогах уже есть сценарии, и все ненужные можно убрать, чтобы излишне не нагружать систему, или запланировать их выполнение на другое время.

Чтобы просмотреть список существующих заданий, выполните команду `crontab` с параметром `-l`. Для редактирования уже созданных записей выполните команду `crontab` с параметром `-e`. В ответ на это запустится текстовый редактор, в котором можно корректировать записи.

Кстати, если вы не работали с этим редактором, то могут возникнуть проблемы, потому что он достаточно специфичный. Тогда нажмите клавишу `<F1>` и воспользуйтесь справкой. Команды тоже вводятся необычно. По выходе из этого редактора

Если вы не запускаете планировщики `at` и `cron`, то я рекомендую убрать сервис `crond` из автозапуска, а лучше даже вовсе удалить его с сервера. Вы не сможете контролировать то, чем не пользуетесь, потому что не будете обращать на это внимание.

В каталоге `/etc` имеется файл `crontab`, в котором находятся все настройки сервиса `cron`. Я рекомендую внести в начало этого файла следующую директиву:

```
CRONLOG=YES
```

Это позволит записывать в журнал команды, которые выполнялись в `cron`. Журнал сервиса находится в файле `/var/cron/log`. Если что-то произойдет без вашего ведома, то это можно будет определить по журналу.

3.6. Настройка сети

В Linux сеть является одним из основных компонентов. Даже если у вас нет подключения к Интернету, будет установлено локальное сетевое окружение, и сетевые функции станут использоваться некоторыми сервисами.

Графическая система Linux построена по клиент-серверной архитектуре. Журналирование также построено по сетевому принципу. Если запустить программу `ifconfig`, то она покажет хотя бы одно работающее сетевое устройство с адресом `127.0.0.1`. Этот интерфейс используется системой для общения с самой собой. Плохо, когда говорит сам с собой человек, а когда это делает компьютер — это нормально и даже хорошо.

Ubuntu серверной версии перестал включать эту утилиту в установку по умолчанию, потому что она устарела, поэтому ее придется поставить командой:

```
sudo apt install ethtool
```

Более современная версия программы просмотра сетевых устройств — `ip` — будет описана в разд. 3.6.4.

После установки ОС Linux легко определяет сетевые карты. С этим у меня еще с начала XXI века проблем не было. Но для работы в сети этого недостаточно. Во время инсталляции вы уже могли указать основные параметры подключения, но иногда появляется необходимость изменить настройки, и не будете же вы переустанавливать систему, когда нужно выполнить всего пару команд.

Для передачи данных по сети необходимо установить и настроить протокол — правила, по которым два удаленных устройства будут обмениваться информацией. Правила описывают, нужно ли устанавливать соединение, как происходит проверка целостности переданных данных, требуется ли подтверждение доставки и т. д. Все это реализовано в протоколе, а ваша задача — лишь правильно его настроить.

3.6.1. Адресация

Основным для Linux является ставший стандартом для Интернета протокол TCP/IP (Transmission Control Protocol / Internet Protocol, протокол управления передачей /

А вот такая — ошибочной (справа от нулей не может быть единиц):

```
11111111.11111111.00000000.11111111
```

Получается, что первые три октета в IP-адресе при маске 255.255.255.0 — это адрес сети, а последний — номер компьютера в этой сети, а так как под него в данном случае отведено одно число, максимальное значение которого 255, то нетрудно догадаться, какое количество компьютеров может быть в вашей сети.

Рассмотрим еще пример:

- 192.168.001.001 — IP-адрес;
- 255.255.000.000 — маска подсети.

Здесь первые два октета — это номер сети, а оставшиеся два — номер компьютера в сети. Число, которое можно задать двумя группами, гораздо больше чем 255, а значит, и сеть будет масштабнее.

Теперь можно сделать одно заключение: компьютеры, имеющие один адрес сети (совпадают три первые числа), могут общаться между собой. А вот машины в разных сетях не видят друг друга, и, чтобы они смогли взаимодействовать, необходимо специальное устройство (маршрутизатор), которое объединяет различные сети и может передавать пакеты между ними.

3.6.2. Информация о сетевых подключениях

Для получения информации о текущей настройке сетевых карт и протокола TCP/IP необходимо выполнить команду `ifconfig`. Пример ее вывода можно увидеть в листинге 3.1.

Листинг 3.1. Информация о конфигурации и состоянии сети

```
eth0      Link encap:Ethernet  HWaddr 00:03:FF:06:A4:6C
          inet addr:192.168.77.1 Bcast:192.168.77.255 Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:108 errors:0 dropped:0 overruns:0 frame:0
          TX packets:104 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:7687 (7.5 Kb)  TX bytes:14932 (14.5 Kb)
          Interrupt:11 Base address:0x2000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:122 errors:0 dropped:0 overruns:0 frame:0
          TX packets:122 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:9268 (9.0 Kb)  TX bytes:9268 (9.0 Kb)
```


- ❑ IP-адрес — если вы хотите изменить IP-адрес, то укажите его новое значение в качестве параметра. Например, если нужно поменять текущий адрес на 192.168.77.3, то выполните команду:

```
ifconfig eth0 192.168.77.3
```

Можно одновременно изменить и маску сети. Для этого выполняем директиву:

```
ifconfig eth0 192.168.77.3 netmask 255.255.0.0
```

Здесь после ключевого слова `netmask` показана новая маска сети.

Если в момент изменения адреса сетевой интерфейс отключен, то его можно сразу же запустить командой:

```
ifconfig eth0 192.168.77.3 netmask 255.255.0.0 up.
```

Это основные возможности программы `ifconfig`, с которыми вам придется сталкиваться в реальной жизни. Более подробную информацию можно получить, выполнив команду `man ifconfig`.

3.6.4. Утилита `ip`

Подготавливая это издание, я уже задумался о том, чтобы удалить предыдущие два раздела, но ограничился только написанием этого. Дело в том, что утилита `ifconfig` устарела, и из некоторых дистрибутивов ее уже начали убирать. Убрали ее как раз из CentOS и Ubuntu, так что информация о `ifconfig` в этом издании осталась только на случай, если вы работаете со старой версией дистрибутива, и в следующем издании я, скорее всего, ее уберу совсем.

Для получения информации об IP-адресах используется утилита `ip`. Логично, не правда ли? То же самое, что делает `ifconfig`, мы можем узнать, выполнив команду:

```
ip addr
```

Если выполнить эту команду с параметром `link`, можно увидеть состояние сетевых интерфейсов:

```
ip link
```

В моем случае результат был получен такой:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
   mode DEFAULT group default qlen 1000
2: enp0s3 <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
   state UP mode DEFAULT group default qlen 1000
```

Второй интерфейс — это сетевой интерфейс виртуальной машины, который работает как сетевая карта.

Добавить IP-адрес к сетевому интерфейсу можно с помощью параметров `addr` `ad`:

```
ip addr add 10.0.2.10/24 dev enp0s3
```

Команду эту нужно запускать с правами администратора. Параметр `dev` указывает на то, к какому сетевому интерфейсу я хочу добавить IP-адрес (в моем случае это `enp0s3`).

3.6.6. Протокол IPv6

О протоколе IPv6 заговорили очень давно. Просто текущая версия интернет-протокола IPv4 максимально поддерживает четыре миллиарда уникальных адресов, но этого в действительности уже не хватает. На планете Земля сейчас живет более семи миллиардов человек. Даже если у каждого человека будет по одному устройству, подключенному к Интернету, всем адресов не хватит.

Да, не у всех есть устройства, работающие с сетью, но много людей, имеющих смартфон, планшет и еще хотя бы один компьютер. Веб-сайты в Интернете также нуждаются в адресах. Если бы всем раздавали уникальные интернет-адреса, то их уже давно не хватило бы на всех. Вместо этого целые сети и сайты делят один и тот же IP-адрес между собой.

Работать над новой версией интернет-протокола начали уже очень давно. Сейчас взглянул в Википедию, и там говорится, что первая версия нового IPv6 появилась еще в 1998 году. Как быстро время летит... А стандартом этот новый протокол стал только в 2017-м. И несмотря на наличие стандарта, переход на IPv6 происходит слишком медленно, но он идет. И если он закончится, то всем хватит уникальных адресов, даже каждому чайнику и кофеварке.

С технической точки зрения IPv4 использует 32-битный адрес, а 6-я версия — 128-битный. Если в привычной нам системе IPv4 адрес состоит из 4 чисел от 0 до 255, разделенных точками, то в новом стандарте — 8 групп шестнадцатеричных чисел, разделенных двоеточиями. Взглянем на простой пример:

```
fe80:0db8:0000:0000:0000:ff00:0042:8329
```

Здесь аж три блока, в которых все четыре числа равны нулю, — в таком случае эту группу блоков можно опустить и записать адрес так:

```
fe80:0db8::ff00:0042:8329
```

Обратите внимание, что все три нулевые группы удалены, и по такой записи можно догадаться, что недостающие группы — как раз нулевые. А если у нас вот такой адрес?

```
fe80::ff00::8329
```

Здесь дважды присутствует двойное двоеточие, и как узнать, сколько групп опущено в первом случае и сколько во втором? Из 8 полагающихся групп мы видим только три, значит, здесь было 5 нулевых. Все очень просто — в IPv6-адресе может быть только одно сокращение в виде двойных кавычек, так что адрес с двумя сокращениями некорректный.

В старом стандарте IPv4 для указания локального компьютера используется адрес 127.0.0.1. В IPv6 ту же самую функцию выполняет адрес ::1. И так как вначале идут два двоеточия, значит, перед 1 целых семь групп из нулей — ведь всего групп должно быть 8. Вспоминаем команду `ip addr`, в результате выполнения которой для `lo` интерфейса `lo` в выводе будет два адреса:

```
inet 127.0.0.1/8 scope host lo
inet6 ::1/128 scope host
```


В ответ на это мы увидим примерно следующее:

```
filename: /lib/modules/2.4.18-5asp/kernel/fs/ext3/ext3.o
description: "Second Extended Filesystem with journaling extensions"
author:      "Remy Card, Stephen Tweedie, Andrew Morton, Andreas Dilger,
             Theodore Ts'o and others"
license:     "GPL"
parm:        do_sync_supers int, description "Write superblocks
             synchronously"
```

Таким образом, нам становятся известными имя и расположение файла, его описание, автор, лицензия и т. д. Количество отображаемой информации сильно зависит от модуля, и, если честно, в некоторых случаях она настолько скудна, что предназначение модуля остается непонятным;

- `modprobe` — эта команда в основном используется системой для загрузки установленных модулей, но можно это делать и самостоятельно. В качестве единственного параметра нужно передать команде имя модуля, который надо загрузить.

Например, следующая команда загружает модуль `iptable_nat` (о нем мы будем говорить в *разд. 4.12*):

```
modprobe iptable_nat
```

- `rmmod` — эта команда выгружает модуль, имя которого указано в качестве параметра. Если вы воспользовались модулем для выполнения определенных действий, то не забудьте по окончании работы его выгрузить.

3.8. Переменная *\$PATH*

В Linux имеется переменная `$PATH`, определяющая папки, в которых нужно искать исполняемые программы. Чтобы просмотреть текущее значение этой переменной, нужно выполнить команду:

```
echo $PATH
```

В результате вы должны увидеть список папок, разделенных двоеточиями:

```
/usr/local/bin:/bin:/usr/bin
```

Соответственно, когда мы отдаем в командной строке команду `ls`, то система находит ее исполняемый файл в каталоге `/bin` и запускает его.

При запуске программ из командной строки Linux нужно обязательно указывать `./` перед именем программы или скрипта. Например, допустим, что в текущем каталоге есть исполняемый файл `publish` — тогда для его запуска нужно в консоли выполнить следующую команду:

```
./publish
```

В Windows и в Linux точка указывает на текущий каталог, а, значит, приведенная команда просит запустить `publish` именно из текущего каталога. Впрочем, в Win-



ГЛАВА 4

Управление доступом

Каждый пользователь должен работать в системе под своей учетной записью. Это позволит ему обезопасить свои файлы от чужого вмешательства, по системным журналам определить, когда и кем были произведены некорректные действия, и т. д.

Обычному пользователю вы должны выделять ограниченные права, которые позволяют ему выполнять только необходимые действия. Кроме того, следует минимизировать количество пользователей с большими привилегиями. Некоторые считают, что в Linux только один администратор root, но это не так. Root — это лишь имя, а администраторов может быть очень много.

Если вы являетесь администратором на предприятии, то должны следить за актуальностью записей. При увольнении сотрудника необходимо удалить его учетную запись, чтобы — недовольный увольнением — он не уничтожил важные данные.

Я в своей недолгой работе администратора уже встречался с такими фактами.

Для работы с командами управления доступом вы должны обладать правами администратора, поэтому входить в систему вам надо как пользователю root или использовать команду `su` или `sudo`.

А теперь перейдем к сути проблемы.

4.1. Права доступа

Давайте вспомним команду `ls -al`. Она возвращает список файлов в следующем виде:

```
drwx----- 3 Flenov  FlenovG  4096 Nov 26 16:10 .
drwxr-xr-x  5 root    root    4096 Nov 26 16:21 ..
-rwxr-xr--  1 Flenov  FlenovG   24 Nov 26 16:10 test
```

Как мы уже знаем, первая колонка (занимает 10 символов) — это права доступа. Давайте рассмотрим, из чего она состоит. Первый символ указывает на тип записи. Здесь может находиться одно из следующих значений:

- дефис (-) — обычный файл;
- буква d — каталог;
- буква l — символьная ссылка;
- буква s — сокет;
- буква p — файл FIFO (First In, First Out — первый вошел, первый вышел).

Далее в колонке прав доступа в каждой строке следуют три группы по три символа, определяющие права доступа для различных категорий пользователей:

- первая тройка — владельцу файла;
- вторая — пользователям, входящим в группу владельца;
- последняя — всем остальным.

Каждая такая группа состоит из трех символов: r (чтение), w (запись) и x (выполнение). Наличие буквы говорит о разрешении соответствующего действия.

Вернемся к нашему примеру. Первая строка содержит права доступа: drwx----- . Первый символ d — значит, это каталог. Потом идут три символа rwx — т. е. хозяин файла может читать, записывать и исполнять каталог. Вместо остальных шести символов стоят знаки дефиса — значит, ни у пользователей группы FlenovG, ни у всех остальных никаких прав нет.

Вторая строка: drwxr-xr-x. Это снова каталог. Далее следует тройка символов: rwx — значит, владельцу разрешены все операции. Следующая тройка, соответствующая группе, равна: r-x, а стало быть, возможны чтение и исполнение, но не запись. Последняя тройка: r-x — показывает, что всем остальным пользователям также доступны только чтение и исполнение.

Последняя строка в примере содержит права доступа -rwxr-xr-- для файла (первый символ — дефис). Хозяин файла имеет к нему полный доступ (первая тройка: rwx). Пользователи группы могут читать и выполнять файл, но не могут его изменять (вторая тройка: r-x). Все остальные могут только читать файл (последняя тройка: r--).

Права можно задавать и последовательностью нулей и единиц. Если в определенном месте стоит 1 (указан один из символов r, w или x), то операция разрешена, а если 0 (указан дефис) — действие запрещено. Давайте попробуем записать права rwxr-xr-- в виде нулей и единиц. Запишите вместо букв единицы, а вместо дефисов — нули. Должно получиться 111101100. Разобьем эту комбинацию на три части: 111, 101 и 100. Теперь каждую тройку переведем в восьмеричную систему по следующей формуле:

$$\text{Цифра1} * 4 + \text{Цифра2} * 2 + \text{Цифра3}$$

У нас получатся три цифры: 7, 5 и 4, которые можно рассматривать как десятичное число 754. Запомните его, оно нам пригодится при назначении прав на файлы и каталоги. Чтобы вам в дальнейшем было проще регламентировать доступ, предлагаю все возможные варианты значений для отдельного разряда числа:

- ☐ + — добавить;
- ☐ - — удалить;
- ☐ = — заменить новыми (старые значения будут уничтожены).

После этого устанавливается режим доступа:

- ☐ r — чтение;
- ☐ w — запись;
- ☐ x — выполнение;
- ☐ X — выполнение, если файл является каталогом или уже имеет аналогичные права для какого-либо пользователя;
- ☐ s — SUID- или SGID-бит (см. *разд. 4.5*);
- ☐ t — sticky-бит. В этом случае только владелец файла и каталога сможет удалить его (см. *разд. 4.4*);
- ☐ u — всем пользователям, как и у владельца;
- ☐ g — всем пользователям, как и у группы;
- ☐ o — всем пользователям, как и у остальных пользователей.

В числовом представлении команда выглядит следующим образом:

`chmod права файл`

Права передаются в виде восьмеричного числа из четырех разрядов:

- ☐ первый разряд определяет дополнительный бит и может принимать одно из значений:
 - 1 — бит принадлежности;
 - 2 — бит SGID;
 - 4 — бит SUID.

Если дополнительные биты устанавливать не надо, этот разряд можно опустить;

- ☐ второй разряд определяет права пользователя — это может быть число от 0 до 7 включительно;
- ☐ третий разряд определяет права группы — это снова может быть число от 0 до 7;
- ☐ четвертый разряд определяет права остальных пользователей. Как вы думаете, какие значения может принимать параметр? Ну конечно же, от 0 до 7.

Например, мы хотим, чтобы владелец и группа имели все права (число 7) на файл `text`, а остальные пользователи могли его только выполнять (число 1).

Команда будет выглядеть следующим образом:

`chmod 771 text`

Число 771 в символьном виде соответствует правам: `rwXrwx--x`.

Следующая команда отменит возможность чтения файла у группы:

`chmod g-r text`

умолчанию должно быть запрещено все. Открываем доступ только на то, что необходимо, и не даем лишних прав.

Очень часто в Интернете можно увидеть вопросы типа: не могу сделать что-то из-за недостатка прав, и я много раз видел рекомендации установить права 777, чтобы решить проблему. Не уверен, что это лучшее решение.

В принципе в установке по умолчанию сейчас уже многое, подлежащее запрещению, запрещено, поэтому простые пользователи не могут просмотреть или изменить важные системные файлы и вынуждены пользоваться командой `sudo` для выполнения каких-либо системных операций.

4.1.4. Права по умолчанию

Когда пользователь создает новый файл или каталог, то им назначаются права по умолчанию. Давайте рассмотрим это на примере. Для создания файла выполним команду `ls` и перенаправим вывод в файл:

```
ls -al >> testfile
```

Теперь проверим права на этот файл с помощью команды `ls -al`. Должно получиться: `-rw-r--r--`, т. е. владелец может читать и изменять файл, а пользователи его группы и все остальные — только просматривать. В старых системах и в некоторых дистрибутивах права могут оказаться и такими: `-rw-rw-r--`, т. е. пользователи группы тоже смогут корректировать файл. Но в любом случае все получают возможность читать файл — это разрешено. Такие права нарушают главное правило безопасности.

В этом отношении политика назначения прав по умолчанию может быть не совсем эффективна — если вы создадите файл, который хранит конфиденциальные данные, информация из него будет доступна для всеобщего обозрения. И если вы захотите понизить права, то любой сможет увидеть и прочитать этот файл.

Ситуацию можно изменить, если понимать, как создаются права для нового файла. Они рассчитываются на основе маски, текущее значение которой определяется командой `umask`. Если выполнить ее после установки системы, будет получено значение 0022 или 002.

Посмотрим, как маска влияет на регламентацию доступа. По умолчанию права для файлов устанавливаются в значение 666 минус маска, а для каталогов — 777 минус маска.

Теперь ясно, что если маска равна 002, то для нового файла будут установлены права $666 - 002 = 664$, а это соответствует: `-rw-rw-r--`. При значении маски 0022 формула изменится на $666 - 0022 = 644$, что будет означать: `-rw-r--r--`.

Для каталогов расчет аналогичный, и при маске, равной 002, по умолчанию будут устанавливаться права $777 - 002 = 775$ (`drwxrwxr-x`). В случае с маской 0022 значение определяется как $777 - 0022 = 755$, а это соответствует правам: `drwxr-xr-x`, т. е.

ВНИМАНИЕ!

Если вы решили использовать символьные ссылки, то всегда помните особенность формирования прав доступа на файлы. Можете даже выбить на мониторе надпись: «Ссылки на файлы создаются с полными правами!!!»

4.1.6. Права доступа к ссылкам

В Linux изначально была реализована слишком простая система прав доступа, унаследованная у UNIX-систем. Обеспечить эффективное управление правами доступа с помощью всего трех наборов прав весьма проблематично. С точки зрения безопасности подход на основе ACL (Access Control List, список управления доступом), который реализован в Windows, намного эффективнее.

В Linux списки доступа появились относительно недавно. Долгое время для этой ОС имелись нестандартные решения, которые были сложными в управлении, и требовалось накладывать специальные патчи на ядро, чтобы реализовать поддержку списков. Теперь все работает «из коробки».

Давайте создадим какой-нибудь файл, на котором будем пробовать управление правами доступа с помощью списков:

```
echo Hello > test.txt
```

Эта команда выводит просто слово **Hello**, а с помощью символа > этот вывод перенаправляется в файл test.txt. Теперь у нас есть файл, в котором просто одна строка с одним словом: Hello.

Посмотрим на списки доступа к этому файлу, для чего выполним команду `getfacl`:

```
getfacl test.txt
```

В результате вы должны увидеть что-то типа:

```
# file test.txt
# owner mflenov
# group mflenov
user::rw-
group::rw-
other::r--
```

Это все очень похоже на то, что мы уже видели в этой главе ранее. Здесь нам в форме комментариев ко второй и к третьей строкам сообщают, какой пользователь является владельцем и какая группа имеет доступ.

Два двоеточия, разделяющие имя параметра и права, не случайны. Пустота между ними указывает на то, что это права владельца. Другими словами, эти же права можно было бы записать так:

```
user:mflenov:rw-
group:mflenov:rw-
```

Просто, когда указываются права доступа владельца и группы владельца, это имя система не показывает.


```
user::rw-  
user:misha:rwx  
group::rw-  
group:economic:rw-  
mask::rwx  
other::r--
```

Чтобы отменить права доступа для пользователя, меняем ключ `-m` на `-x` и просто указываем имя пользователя:

```
setfacl -x misha test.txt
```

Чтобы сбросить список доступа к параметрам по умолчанию, выполняем команду с ключом `-b`:

```
setfacl -b misha test.txt
```

4.2. Управление группами

Что такое группы? Допустим, что в вашей сети 1000 пользователей, 500 из которых должны иметь доступ к файлам бухгалтерской отчетности. Как поступить? Можно каждому из 500 пользователей назначить права на нужный файл и забыть об этом до определенного времени. А теперь представим, что нужно отменить это разрешение. Опять выполнять 500 команд для каждого файла? А может быть, писать собственную программу, выполняющую такую отмену? Оба способа неудобны и требуют больших усилий.

В Linux нельзя просто так дать пользователю право на файл — это можно только в Windows с использованием списков ACL. Что-то похожее пытаются реализовать в Linux (см. *разд. 4.1.6*), но пока подобные реализации не получили популярности.

Вместо этого можно объединить многих пользователей в группу, а уже ей дать право на использование определенного файла. Впоследствии, если нужно будет запретить доступ, достаточно одной команды, чтобы отключить разрешение для группы, и все 500 пользователей больше не смогут работать с файлом. Удобно? Даже очень.

Но тут возникает другая проблема — а если пользователю нужно быть сразу в двух группах? Снова натываемся на барьер старой системы прав Linux.

ПРИМЕЧАНИЕ

В ОС Red Hat Linux все пользователи приписываются к какой-либо группе. Если при введении новой учетной записи группа не указана, то она будет создана по умолчанию под именем пользователя.

4.2.1. Добавление группы

Для создания новой группы служит команда `groupadd`, синтаксис ее такой:

```
groupadd [-g gid [-o]] [-r] [-f] имя
```


4.2.2. Редактирование группы

Для редактирования параметров группы можно напрямую изменять файл `/etc/group`, но я рекомендую использовать команду `groupmod`. У этой команды такие же ключи, что и у `groupadd`, но она не добавляет группу, а изменяет параметры уже существующей.

4.2.3. Удаление групп

Для удаления группы служит команда `groupdel`:

```
groupdel имя
```

При выполнении этой команды вы должны самостоятельно проверить все файлы, владельцем которых является удаляемая группа, и при необходимости изменить собственника, иначе к таким файлам сможет получить доступ только администратор.

Надо еще заметить, что группу нельзя удалить, если в ней есть пользователи. Сначала их нужно вывести из группы и только потом выполнять команду `groupdel`.

4.3. Управление пользователями

Для добавления пользователя служит команда `useradd`. С ее помощью также можно изменить значения по умолчанию, которые будут присваиваться учетной записи.

Команда `useradd` выглядит следующим образом:

```
useradd параметры имя
```

Параметров очень много, большинство из них вам знакомо по файлу `/etc/passwd`, который мы рассматривали в *главе 3*. Вот наиболее часто используемые аргументы:

- ☐ `-c` комментарий — простое текстовое описание, которое может быть любым;
- ☐ `-d` каталог — домашний каталог пользователя;
- ☐ `-e` дата — дата отключения учетной записи, после которой пользователь станет неактивным, вводится в формате ГГГГ-ММ-ДД;
- ☐ `-f` число — количество дней до отключения записи навсегда. Если указать 0, то учетная запись будет считаться некорректной сразу после устаревания пароля;
- ☐ `-g` группа — основная группа, которой будет принадлежать пользователь. Можно указывать как имя, так и идентификатор (напомню, что в Red Hat каждый пользователь принадлежит какой-либо группе);
- ☐ `-G` группа[, ...] — дополнительные группы, в которые будет включен пользователь. Имена групп указываются через запятую;
- ☐ `-m` — ключ для создания домашнего каталога пользователя. В созданный каталог будут скопированы все файлы из каталога `/etc/skel`;

3 of these 4 classes, or a 7 character long password containing characters from all the classes. Characters that form a common pattern are discarded by the check. A passphrase should be of at least 3 words, 12 to 40 characters long and contain enough different characters.

Alternatively, if noone else can see your terminal now, you can pick this as your password: "trial&bullet_scare".

Что по-русски звучит примерно так:

Изменяется пароль для пользователя robert.

Сейчас вы можете выбрать новый пароль или идентификационную фразу.

Хороший пароль должен состоять из заглавных и строчных букв, цифр и других знаков. Вы можете ввести пароль длиной в 8 символов с использованием значений как минимум 3 из 4 указанных классов или пароль из 7 символов, сочетающий знаки из всех классов. Пароли, содержащие часто используемые шаблоны, будут отвергнуты.

Идентификационная фраза должна состоять из 3 слов общей длиной от 12 до 40 символов и содержать разнообразные знаки.

В качестве альтернативы, если в данный момент никто, кроме вас, не смотрит на ваш терминал, можно использовать пароль trial&bullet_scare.

Как видите, команда `passwd` знакомит нас с основными правилами создания сложных паролей и даже предлагает пример, который достаточно длинный и содержит различные символы. Но я не стал бы его использовать, потому что он состоит из вполне читаемых слов. Можно запустить подбор паролей по словарю, где различные слова объединяются, как это делает `passwd`. Такая процедура займет значительно больше времени, чем подбор пароля из одного слова, но все же намного меньше, чем подбор шифра вроде OLhslu_9&Z435drf. Для нахождения этого пароля словарь не поможет, а полный перебор всех возможных вариантов займет весьма много времени.

А давайте-ка посмотрим, что сейчас находится в домашнем каталоге нового пользователя. Вы думаете, что там ничего нет? Проверим. Перейдите в каталог `/home/robert` или выполните команду:

```
ls -al /home/robert
```

Ключ `-a` заставляет отображать все файлы (в том числе и системные), а ключ `-l` выводит подробную информацию. Результат выполнения такой команды должен выглядеть примерно следующим образом:

```
drwx----- 3 robert  robert  4096 Nov 26 16:10 .
drwxr-xr-x  5 root    root    4096 Nov 26 16:21 ..
-rw-r--r--  1 robert  robert    24 Nov 26 16:10 .bash_logout
-rw-r--r--  1 robert  robert   191 Nov 26 16:10 .bash_profile
-rw-r--r--  1 robert  robert   124 Nov 26 16:10 .bashrc
-rw-r--r--  1 robert  robert  2247 Nov 26 16:10 .emacs
-rw-r--r--  1 robert  robert   118 Nov 26 16:10 .gtkr
drwxr-xr-x  4 robert  robert  4096 Nov 26 16:10 .kde
```

Обратите внимание, что в каталоге содержатся пять файлов и один каталог. Самое интересное находится в третьей и четвертой колонках вывода, где располагаются

некоторые параметры, в том числе и маску. В *разд. 4.1* мы говорили об излишних правах, которые назначаются всем новым файлам пользователя, и научились понижать их с помощью команды `umask`.

Зайдите под учетной записью `robert` и посмотрите ее маску с помощью команды `umask`. Обратите внимание — она равна `0002`. То есть мы в *разд. 4.1* изменили свою маску, но `robert` при создании получил другую, которую надо изменить. Если вы забудете это сделать, то могут возникнуть проблемы. Чтобы этого не произошло, я рекомендую добавить в конец файла `.bash_profile` строку:

```
umask 0077
```

Лучше всего это сделать в файле `/etc/skel/.bash_profile`, потому что он копируется во все папки новых пользователей, и можно быть уверенным, что все они получат нужную маску.

Для повышения безопасности я не рекомендую присваивать каталогам имена учетных записей. При добавлении пользователя `robert` по умолчанию для него будет создан каталог `/home/robert`. Такое соответствие может сыграть злую шутку. Если злоумышленник узнает имя каталога, то он легко сможет определить имя пользователя, которому он принадлежит, и наоборот. Впрочем, я бы не назвал эту проблему большим упущением в плане безопасности, но все же любая предсказуемость ее ухудшает.

Поэтому при создании пользовательских каталогов достаточно даже просто добавить к имени какой-либо префикс, и это уже может усложнить задачу злоумышленнику.

4.3.2. Изменение настроек по умолчанию

Давайте теперь посмотрим, откуда берутся значения по умолчанию. Все это хранится в файле `/etc/default/useradd`. Взглянем на содержимое этого файла:

```
# useradd defaults file
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
```

Номер `100` присваивается пользовательской группе, у которой по умолчанию очень мало прав. Это как гостевой пароль, который позволяет только просматривать файлы.

Файл `/etc/default/useradd` можно редактировать вручную или воспользоваться командой `useradd` для изменения значений по умолчанию. Чтобы внести изменения с ее помощью, нужно сразу после имени команды указать ключ `-D`. После этого могут идти следующие опции:

- ❑ `-g` — изменить группу;
- ❑ `-b` — установить домашний каталог;

Если для пользователя была автоматически создана группа и в ней никого больше нет, то можно удалить и ее, выполнив команду `groupdel`.

4.3.5. Настройка процедуры добавления пользователей

Для полного понимания процесса создания учетных записей нам нужно познакомиться еще с файлом `/etc/login.defs`. В нем хранятся настройки, которые будут использоваться при добавлении пользователей. Содержимое файла можно увидеть в листинге 4.1.

Листинг 4.1. Файл `/etc/login.defs`

```
# *REQUIRED*
#   Directory where mailboxes reside, _or_ name of file, relative to the
#   home directory.  If you _do_ define both, MAIL_DIR takes precedence.
#   QMAIL_DIR is for Qmail
#
#QMAIL_DIR      Maildir
MAIL_DIR        /var/spool/mail
#MAIL_FILE      .mail

# Password aging controls:
#
#PASS_MAX_DAYS  Maximum number of days a password may be used.
#PASS_MIN_DAYS  Minimum number of days allowed between password changes.
#PASS_MIN_LEN   Minimum acceptable password length.
#PASS_WARN_AGE  Number of days warning given before a password expires.
#
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_MIN_LEN     5
PASS_WARN_AGE   7

#
# Min/max values for automatic uid selection in useradd
#
UID_MIN         500
UID_MAX         60000

#
# Min/max values for automatic gid selection in groupadd
#
GID_MIN         500
GID_MAX         60000
```


К сожалению, изменение этих параметров не влияет на уже существующих пользователей.

Чтобы просмотреть или изменить политику паролей для существующего пользователя, нам понадобится утилита `chage`. Для просмотра текущих параметров пользователя выполняем эту команду с ключом `-l` и указываем имя интересующего нас пользователя:

```
chage -l mflenov
```

Результатом выполнения команды будет таблица вида:

```
Last password change           : never
Password expires               : never
Password inactive              : never
Account expires                : never
Minimum number of days between password change : 0
Maximum number of days between password change  : 99999
Number of days of warning before password expires. : 7
```

Давайте поменяем эти значения на более рациональные:

```
chage -m 0 -M 90 -W 10 mflenov
```

А теперь поподробнее:

- ☐ ключ `-m` — меняет минимальное значение дней — я оставил 0;
- ☐ ключ `-M` — максимальное количество дней устанавливаем в рекомендованные 90;
- ☐ ключ `-W` — количество дней до истечения пароля, после чего система начнет показывать предупреждение. Просто для примера решил поменять на 10, хотя недели вполне достаточно.

Еще раз вызовите команду с параметром `-l` — чтобы убедиться, что изменения отразились на пользователе.

Более подробно о команде `chage` мы будем говорить в *разд. 14.1*.

4.3.6. Взлом паролей

Я еще раз хочу напомнить о недопустимости использования простых паролей не только администратором, но и всеми пользователями системы. К уязвимостям, которые находят в Linux-системах, относятся, в частности, эксплойты, которые позволяют повысить права хакера от простого пользователя до `root`. Но если взломщик не сможет получить доступ даже в качестве простого пользователя, то и воспользоваться таким эксплойтом ему будет невозможно.

Сложные пароли должны быть абсолютно у всех пользователей. Вспомните, как хранятся пароли. Они зашифрованы необратимым образом. Это значит, что при простом переборе каждый возможный вариант тоже шифруется, а потом сравнивается с зашифрованным паролем из файла `/etc/shadow`. Поскольку такое шифрование

Попробуйте выполнить следующие команды:

```
su root
ls -al >> /home/flenov/1.txt
ls -al /home/flenov/1.txt
su flenov
rm /home/flenov/1.txt
```

В первой строке переключаемся под пользователя `root`, чтобы от его имени выполнять команды. Вторая команда отображает файлы из текущего каталога, а результат сохраняет в файл `/home/flenov/1.txt`. Каталог выбран не случайно — это домашний каталог пользователя `flenov`, в котором он может делать все что хочет.

Следующая строка применяет команду `ls -al` к только что созданному файлу, чтобы просмотреть о нем подробную информацию. Убедитесь, что владелец `root` и права доступа равны: `-rwxr--r--`. Если это не так, то запретите изменение файла для группы и для всех. В принципе достаточно запретить изменение для всех, потому что пользователь `flenov` в группу `root` не входит, а значит, не сможет изменять файл.

Теперь переключаемся на пользователя `flenov` и удаляем файл. Прикол в том, что файл удалится. Единственное, что вы увидите, — предупреждение, что удаляется файл, защищенный от изменений.

И чтобы этого не произошло, вы должны ограничивать доступ не только к файлам, но и каталогам.

Бывают случаи, когда каталог должен иметь все права. Как правило, это открытые папки, через которые пользователи могут обмениваться файлами. Но при этом нужно защититься таким образом, чтобы только администратор или владелец файла могли его удалить. Все остальные пользователи не должны иметь прав на уничтожение уже существующих чужих файлов. Как же решить эту проблему, когда каталог необходимо сделать доступным всем, а его содержимым должны управлять только хозяева?

Допустим, у вас есть каталог `shared`. Для того чтобы в нем могли удалять файлы только владельцы, нужно установить для него `sticky`-бит. Это делается командой `chmod` с параметром `+t`:

```
chmod +t shared
```

Попробуйте выполнить команду `ls -al` и посмотреть права доступа к каталогу. Вы должны увидеть: `drwxrwxrwt`. Обратите внимание, что на месте символа `x` в области прав для всех пользователей стоит `t`. Это и указывает на установленный `sticky`-бит. Теперь попробуйте удалить из этого каталога файл, принадлежащий другому пользователю. Вы увидите сообщение: **rm: cannot unlink 'имя файла': Operation not permitted**.

В Linux есть каталог `/tmp`, для которого как раз установлены права `drwxrwxrwx` и в котором сохраняются временные данные всех пользователей. В современных дистрибутивах на этот каталог уже выставлен `sticky`-бит.

Просмотреть текущие атрибуты можно с помощью команды `lsattr`:

```
lsattr filename.txt
```

```
----- filename.txt
```

Первая строка демонстрирует использование команды, а во второй — отображается результат ее работы. Как видите, мы получили набор дефисов вместо атрибутов, а значит, ни один из них не определен.

Для установки атрибута применяется команда `chattr`:

```
chattr атрибуты файл
```

Если требуется рекурсивное изменение доступа к каталогу и ко всем содержащимся в нем файлам и подкаталогам, можно использовать ключ `-R`. В этом случае вместо имени файла укажите в команде `chattr` каталог.

Вот перечень основных атрибутов, применяемых в команде `chattr`:

- `A` — не создавать метку `atime` записи времени последнего обращения к файлу. С точки зрения безопасности этот атрибут несет отрицательный эффект, потому что по дате обращения можно контролировать, когда файл был модифицирован. Поэтому не рекомендую устанавливать этот флаг. Но если у вас под ОС Linux работает личный компьютер и нет необходимости отслеживать историю изменений, то можно установить этот атрибут и тем самым уменьшить количество обращений к диску (отменить лишнюю операцию записи при сохранении файла);
- `a` — позволяет открывать файл только в режиме добавления. Это значит, что уже существующие данные изменить или удалить будет нельзя;
- `d` — заставляет игнорировать файл при копировании. Этот флаг позволяет уменьшить размер резервной копии, но устанавливать его нужно только на файлы, не имеющие ценности и важности, — например, временные;
- `i` — запрещает выполнение с файлом каких-либо действий по коррективке (изменение, удаление, переименование, создание ссылок);
- `s` — делает невозможным восстановление файла после удаления. При удалении файла все пространство на диске, где он был записан, будет заполнено нулями;
- `S` — во время изменения файла все действия будут фиксироваться на жестком диске.

Для установки атрибута перед ним нужно поставить знак «плюс», для снятия — знак «минус». Рассмотрим несколько примеров:

```
chattr +i test
chattr +s test
lsattr test
s--i----- test
```

В первой строке мы добавляем атрибут `i`, а значит, запрещаем какие-либо изменения файла. Во второй строке устанавливаем флаг `s`, и теперь при удалении файла можно быть уверенным, что он уничтожен полностью. Команда в третьей строке

Давайте на примере абстрактного сайта **www.sitename.com** посмотрим, как происходило вторжение. При входе на форуме в просмотр какой-нибудь темы в адресной строке браузера появляется ссылка:

<http://www.sitename.com/forum/viewtopic.php?p=5583>

Если к этой ссылке добавить в конец следующий текст:

```
&highlight=%2527.$poster=%60команда Linux%60.%2527
```

то указанная команда Linux выполнится на сервере.

Вот так, например, можно было просмотреть на сервере файлы каталога /etc:

```
&highlight=%2527.$poster=%60ls%09/etc%09-la%60.%2527
```

А вот эта команда могла удалить главную страницу сайта:

```
&highlight=%2527.$poster=%60rm%09index.php%60.%2527
```

Как видите, благодаря ошибке в одной строке скрипта форума под угрозой оказалось существование всего сервера.

В коде форума phpBB после этого находили не одну критическую уязвимость, но я уже с тех пор никогда его не использовал, т. к. пришел к выводу, что лучше написать свой собственный код сайта с минимальными возможностями, но зато безопасный для использования.

А ведь опасность можно уменьшить, ограничив права веб-сервера в ОС. Для этого администраторы должны создать виртуальную среду для выполнения веб-сервиса, сделав при этом все остальные разделы сервера недоступными для злоумышленника. Тогда и каталог /etc окажется недостижимым, и максимальный вред, который сможет нанести злоумышленник, — это уничтожить сайт и нарушить работу веб-сервиса, но все остальные будут трудиться в штатном режиме. Восстановить один сервис проще, чем налаживать абсолютно все.

После случая с форумом phpBB я целый день бродил по Интернету в поисках уязвимых форумов, и таких оказалось много, потому что администраторы сайтов явно не следили за обновлениями. Я думаю, что в скором времени эти администраторы пройдут через все круги ада. Рано или поздно взломщик найдет их форумы, и администраторам нужно будет молиться, чтобы он не уничтожил всю базу, а только пошалил. Хочется еще раз напомнить о необходимости обновления всех программ, сервисов и самой ОС. Если вы сможете исправить ошибку ранее, чем хакер ее найдет, то обезопасите свою жизнь.

Во время поисков уязвимых форумов я просматривал возможность получения доступа к каталогу /etc, чтобы увидеть не только количество неопытных владельцев сайтов, но и некомпетентных администраторов. Вы не поверите, но доступ был открыт, наверное, на 90% серверов. Это безграмотность администраторов или их лень? Точно сказать трудно. Защищены были только крупные серверы.

Этот случай произошел очень давно, и я надеюсь, что сейчас ситуация с безопасностью намного лучше.

4.7.2. Установка Jail

Встроенная в Linux-систему программа `chroot`, создающая на сервере виртуальные пространства, сложна для применения и не очень удобна — приходится выполнять слишком много операций. Именно поэтому администраторы больше любят программу `Jail`, которую можно найти в Интернете по адресу: <http://www.jmcresearch.com/projects/jail/>. Скачайте ее и поместите архив в свой каталог. Для того чтобы его разархивировать, выполните следующую команду:

```
tar xzvf jail.tar.gz
```

В текущем каталоге появится новый каталог `jail` с исходным кодом программы. Да, именно с исходным, потому что код открыт, и программа поставляется в таком виде.

Теперь нужно перейти в каталог `jail/src` (командой `cd jail/src`) и отредактировать файл `Makefile` (например, встроенным редактором `MC`). В самом начале файла идет множество комментариев, и их мы опустим. После этого вы сможете увидеть следующие параметры:

```
ARCH= __LINUX__  
#ARCH= __FREEBSD__  
#ARCH= __IRIX__  
#ARCH= __SOLARIS__
```

```
DEBUG = 0  
INSTALL_DIR = /tmp/jail  
PERL = /usr/bin/perl  
ROOTUSER = root  
ROOTGROUP = root
```

Вначале задается тип ОС — по умолчанию установлена `LINUX`, а следующие три строки для `FreeBSD`, `IRIX` и `Solaris` закомментированы. Оставим это как есть. Что нужно изменить, так это каталог для установки (параметр `INSTALL_DIR`). В последней версии (на момент подготовки этого издания книги) по умолчанию указывается каталог `/tmp/jail`. Не знаю, зачем было так сделано, ведь этот каталог предназначен для временных файлов и должен быть доступен для чтения абсолютно всем. Раньше по умолчанию был прописан каталог `/usr/local`, и именно его я советую здесь указать. Больше ничего менять не надо.

Для выполнения следующих директив вам понадобятся права `root`, поэтому войдите в систему как администратор или получите нужные права, запустив команду `su root`.

Перед компиляцией и установкой убедитесь, что у файла `preinstall.sh` есть права на выполнение. Если нет, воспользуйтесь следующей командой:

```
chmod 755 preinstall.sh
```

Теперь все готово к установке. Находясь в каталоге `jail/src`, выполните команды:

```
make  
make install
```


Лишнее следует удалить, но лучше это делать после того, как вы убедитесь, что все работает.

Итак, необходимые программы перенесены, и окружение готово. Теперь сюда можно установить сервис:

```
/usr/local/bin/addjailsw /home/chroot -P httpd
```

В этом примере в новое окружение устанавливается программа `httpd` и все необходимые ей библиотеки. Программа `jail` сама определит, что будет нужно.

Теперь в новое окружение можно добавлять пользователя. Это выполняется командой:

```
/usr/local/bin/addjailuser chroot home sh name
```

Здесь `chroot` — виртуальный корневой каталог, в нашем случае должно быть указано `/home/chroot`. Параметр `home` — домашний каталог пользователя относительно виртуального каталога. Аргумент `sh` — командный интерпретатор, а `name` — имя пользователя, которое мы хотим добавить (оно должно уже существовать в основном окружении ОС).

Посмотрим, как можно добавить пользователя `robert` (он у нас уже есть) в виртуальную систему:

```
/usr/local/bin/addjailuser /home/chroot \  
/home/robert /bin/bash robert
```

ПРИМЕЧАНИЕ

У меня здесь команда не уместилась в одну строку, поэтому я сделал перенос с помощью символа «\», который обозначает, что директива не закончилась, и ее продолжение в следующей строке.

Если параметры указаны верно, то вы должны увидеть уведомление **Done**, в противном случае будет выведено сообщение об ошибке.

Для запуска сервера `httpd` (в Linux это сервер Apache) в виртуальном окружении должен существовать пользователь `apache`. В реальной оболочке он уже есть. Давайте посмотрим его параметры и создадим такого же:

```
/usr/local/bin/addjailuser /home/chroot \  
/var/www /bin/false apache
```

Как теперь попасть в новое окружение? Выполните команду:

```
chroot /home/chroot
```

и вы окажетесь в новом окружении. Только учтите, что большинство команд здесь не работает. Так, например, мы не установили программу `MC`, поэтому вы не сможете ею воспользоваться.

Чтобы убедиться, что вы находитесь в виртуальном окружении, выполните команду:

```
ls -al /etc
```

Вы увидите всего несколько файлов, которые составляют малую часть того, что доступно в реальном каталоге `/etc`. Можете просмотреть файл `/etc/passwd` — в нем

Впрочем, пользователя `root` можно оставить, но установить ему идентификатор больше нуля. Я иногда создаю учетную запись `root` с `UID=501` или выше и устанавливаю мониторинг за этой записью — вдруг кто-то попытается подобрать пароль или, не дай бог, войти в систему под этим именем. Получив доступ под именем пользователя `root`, взломщик думает, что он обладает всеми правами, но в реальности оказывается простым пользователем.

Итак, злоумышленник, проникнув в систему, может не пользоваться учетной записью `root`, а отредактировать любую другую или добавить еще одну с нулевым идентификатором пользователя и получить максимальные права в системе. Если вы являетесь администратором сервера, то должны отслеживать подобные трансформации и останавливать любые попытки изменения идентификаторов.

Команда `id` позволяет узнать идентификаторы пользователя. Если она вызывается без параметров, то на экран будут выведены идентификаторы текущего пользователя. Чтобы получить информацию о конкретной учетной записи, нужно выполнить команду:

```
id имя
```

Например, давайте посмотрим параметры пользователя `robert`. Для этого выполним команду:

```
id robert
```

Результатом будет строка:

```
uid=501(robert) gid=501(robert) group=501(robert)
```

Таким способом можно в любой момент определить идентификатор пользователя и узнать, какими правами он реально обладает.

4.9. Права приложений

Права доступа могут быть не только у пользователей, но и у приложений. Для управления такими правами в Linux сейчас существует две конкурирующие технологии: SELinux (используется в CentOS) и AppArmor (используется в Ubuntu). Обе технологии включены в состав ядра Linux и выполняют примерно одинаковые функции, но по-разному.

AppArmor определяет файлы и конфигурацию доступа по полному пути. Если создать жесткую ссылку на какой-либо файл, то защита на ссылку уже не сработает. При этом SELinux доступ через жесткую ссылку запретит.

Существуют разные мнения, но большинство все же соглашается, что с точки зрения управления AppArmor проще. Два основных дистрибутива, которые его поддерживают, — это SUSE и Ubuntu.

Начнем знакомство с AppArmor с простой команды `apparmor_status`, которая возвращает статус. В результате выполнения команды вы должны увидеть сообщение, что модуль загружен, и увидеть профили, которые используются. Можно установить свежие профили, выполнив команду:

```
apt-get install apparmor-profiles
```


Простейший способ обхода сетевого экрана — подделка IP-адреса. Например, мне приходилось работать в сети, где простым пользователям запрещалось использовать почтовые протоколы SMTP и POP3 (подключение на 25-й и 110-й порты соответственно). Я относился к этой категории и не мог ни получать, ни отправлять почту, однако у моего начальника такой доступ был. Использование веб-интерфейса для работы с почтовыми сервисами также блокировалось на уровне прокси-сервера. И вот однажды мне очень нужно было срочно отправить письмо. Для этого я выполнил следующее:

1. Дождался, когда начальник выйдет из кабинета.
2. Выключил его компьютер.
3. Сменил свой IP-адрес на установленный на его компьютере.
4. Спокойно отправил почту и вернул свой старый IP-адрес.

Когда начальник вернулся, он решил, что компьютер просто завис (это было еще во времена Windows 98), и ничего не заподозрил, а я без проблем смог воспользоваться сервисом, который был мне запрещен.

Однако такой метод можно реализовать только в локальной сети, а в Интернете подобное уже не пройдет. Поэтому сетевой экран все же является очень хорошим методом защиты от сетевых атак.

В ОС Linux в качестве брандмауэра выступает программа, фильтрующая информацию на основе определенных правил, в которых должно быть четко прописано, какие пакеты могут обрабатываться или отправляться в сеть, а какие нет. Благодаря этому большинство атак захлебнутся уже на входе в компьютер, потому что сетевой экран не позволит сервисам даже увидеть потенциально опасные пакеты.

Сетевой экран может быть установлен на каждом компьютере сети в отдельности (защищать его в зависимости от выполняемых задач) и на входе в сеть (рис. 4.2). Во втором случае компьютер Firewall реализует общие настройки безопасности для всех компьютеров в сети.

Если в вашей сети очень много компьютеров, то управлять ими по отдельности и обновлять политику безопасности каждого становится затруднительным. Установка единого сервера с сетевым экраном позволяет упростить эти процедуры. Лучше всего, если компьютер с сетевым экраном выступает как шлюз или как анонимный прокси-сервер для доступа в Интернет остальных участников сети. В таком случае хакер изначально будет видеть только этот компьютер, а остальные окажутся для него как бы спрятанными за занавеской. Чтобы проникнуть на любую машину в сети, злоумышленник должен будет сначала получить доступ к компьютеру с Firewall. Таким образом, защита целой сети немного упрощается. Подробнее о прокси-серверах вы можете узнать в *главе 9*.

Сейчас большинство пользователей подключается к Интернету через домашние сети Wi-Fi, и даже в их домашних роутерах может работать свой собственный брандмауэр. Доступа к его настройкам, возможно, и не будет — задачей такого сетевого экрана является предоставление базовой безопасности.

- ☐ без предупреждений (deny, запрет);
- ☐ с ссылкой на компьютер отправителя сообщения об ошибке (reject, отклонение).

Я бы не выбирал второй вариант, потому что незачем направлять лишние пакеты. Во-первых, просто не вижу смысла утруждать сеть доставкой этой мелочи. Во-вторых, у хакера не будет ответа на основной его вопрос: сервис закрыт сетевым экраном или просто не включен. Конечно, и это тоже мелочь, но все же зачем нам делать кому-то жизнь легче.

При настройке правил можно использовать два варианта фильтра:

- ☐ разрешено все, что не запрещено;
- ☐ запрещено все, что не разрешено.

Наиболее безопасным методом является второй, потому что всегда следует отталкиваться от запрета. Изначально необходимо запретить абсолютно все, а потом по мере надобности открывать доступ определенным пользователям к необходимым им сервисам. Именно этой политики вы должны придерживаться, когда настраиваете правила для входящих пакетов.

Если двигаться от разрешения, то по умолчанию все позволено. И администратор может забыть или просто не закрыть какой-либо вид доступа.

4.10.2. Параметры фильтрации

Основными параметрами пакета, по которым производится фильтрация, являются номер порта источника или приемника, адрес отправителя или назначения и протокол. Как мы уже знаем, сетевым экраном поддерживаются три базовых протокола: TCP, UDP и ICMP, на основе которых строится все остальное (FTP, HTTP, POP3 и пр.).

Обращаю ваше внимание, что фильтровать нужно пакеты, идущие в обе стороны. Проверка пакетов, приходящих извне, позволяет на самом раннем этапе отсеять любые попытки взломать систему или вывести ее из строя.

А зачем фильтровать то, что уходит из сети? На первый взгляд это бессмысленно, но резон есть, и он достаточно велик. Исходящий трафик могут инициировать и злоумышленники, например:

- ☐ троянские программы, которые могут отправлять в сеть конфиденциальную информацию или сами соединяться с хакером или с его сервером и брать команды из какого-нибудь файла;
- ☐ специализированные программы для обхода правил. Допустим, что вы запретили доступ к определенному порту извне. Хакер может разместить на сервере программу, которая будет перенаправлять трафик с разрешенного порта на запрещенный — наподобие туннелирования OpenSSL (Open Secure Sockets Layer, открытый протокол защищенных сокетов). Профессионалу написать такую утилиту — дело пяти минут;

кроме определенных адресов. Единственный Jump Box-сервер содержит списки доступа по публичным IP-адресам, но этих адресов там не много, и их легко обновлять в случае необходимости.

Протоколы

Протокол TCP используется как базовый для передачи данных по таким протоколам, как HTTP, FTP и др. Запрещать его не имеет смысла, потому что это основа, без которой вы лишитесь всех удобств, предоставляемых нам Всемирной сетью. Для передачи данных по протоколу TCP сначала устанавливается соединение с удаленным хостом, и только потом происходит обмен информацией. Благодаря этому подделка IP-адреса любого участника соединения усложняется, а иногда и становится невозможной.

Протокол UDP находится на одном уровне с TCP, но передает данные без установки соединения. Это значит, что пакет просто посылается в сеть на определенный адрес, и нет гарантии, что он дошел до адресата. Здесь нет никакой защиты от подделки IP-адреса, поэтому в качестве отправителя злоумышленник может указать кого угодно, и наш сервер не увидит подвоха. Если нет особой надобности (ни одна установленная у меня программа не использует UDP), то я запрещаю прохождение таких пакетов в обе стороны.

Протокол ICMP служит для обмена управляющими сообщениями. Через него команда ping проверяет соединение с компьютером, а оборудование или программы сообщают друг другу об ошибках. Если этот протокол использовать только по назначению, то он очень удобен. Но в нашей жизни все далеко от идеала, и ICMP уже не раз становился объектом для DoS-атак. Если обмен управляющими сообщениями необходим используемой вами программе, попробуйте найти другую, но избавьтесь от использования ICMP, который также не требует соединения, потому что использует UDP-запросы.

Фильтрация портов

Первое, на что надо обратить внимание, — это, конечно же, порты. Допустим, что у вас есть веб-сервер, к которому имеют доступ все пользователи. Предположим, что на нем работают абсолютно безопасные сценарии или статические документы HTML. Помимо этого, все программы содержат самые последние обновления и не имеют уязвимостей. Получается, что сервер безопасен? Да, но до поры до времени. Для обновления содержимого необходим какой-то доступ для загрузки файлов — ведь бегать с дисками к веб-серверу никто не будет. Чаще всего для работы с файлами открывают FTP-сервис, а вот это уже дополнительная головная боль.

Для доступа по FTP можно установить наиболее защищенные программы и самые сложные пароли, но хакер рано или поздно может найти вход на этот сервис. Пароль можно подобрать, украсть с компьютера пользователя или вывести пароль у него самыми средствами социальной инженерии. Существуют и другие методы. Любой канал, через который хакер может проникнуть в систему, становится уязвимым, потому что именно его будет взламывать злоумышленник, и как раз на это

Фильтрация нежелательных адресов

Несколько лет назад сервис **www.regnow.com** (который выступает посредником для производителей Shareware-программ, получая деньги от клиентов и обеспечивая безопасность платежей) попытался ограничить доступ с сомнительных IP-адресов. Это вполне логично. Некоторые страны кишат хакерами, и при этом число добропорядочных пользователей программ в них стремится к нулю. К таким государствам отнесли страны Африки и некоторых регионов Восточной Европы, включая развивающуюся Россию.

Ограничительный шаг сервиса оправдывался тем, что в ряде этих стран была весьма сильно развита технология кардинга, когда товар в Интернете заказывается по ворованным кредитным картам. Чтобы исключить кардинг, сервис запретил доступ по целым группам IP-адресов. Впоследствии выяснилось, что обойти эту систему очень просто, — хакеру достаточно было воспользоваться анонимным прокси-сервером в США или Канаде, чтобы проскочить преграду. А вот у добропорядочных пользователей возникли серьезные проблемы, и они лишились возможности использовать сервис для оплаты необходимых услуг.

Из-за столь серьезных недостатков фильтр был снят, и разработчики **regnow.com** больше не пытаются его использовать. Просеивать все возможные прокси-серверы слишком затруднительно, и это дает малый эффект, а репутацию можно потерять навсегда. Так что иногда приходится выбирать между безопасностью и удобством пользования.

Еще пример — я живу в Канаде, и некоторые сервисы из России для нас блокируются. Например, онлайн-телевидение Первого канала блокирует некоторые передачи и фильмы, потому что они получают лицензию для трансляции контента только на территории России. Но у меня есть VPN-подключение (Virtual Private Network), и через серверы в России я могу смотреть любые передачи. Таким же образом люди здесь смотрят фильмы с американского Netflix, в котором больше хороших фильмов, чем в Канаде.

Фильтрация неверных адресов

Очень важно правильно настроить обработку IP-адресов получателя. Пусть сервер настроен таким образом, чтобы в случае прихода неверного пакета отвечать отправителю сообщением о некорректности данных. Проблема заключается в том, что злоумышленник может послать на сервер такой пакет, в котором в качестве отправителя стоит адрес получателя, — т. е. и в том и в другом случае используется IP-адрес сервера. Конечно же, сервис пытался отослать сообщение об ошибке, и отправлял его сам себе, и снова видел ошибочный пакет. Таким образом процесс заклинивал. Если злоумышленник направит тысячи таких пакетов, то сервер только и будет сам себе посылать сообщения об ошибках.

О подобных погрешностях я уже давно ничего не слышал, но нет гарантии, что они не появятся снова. Люди иногда ошибаются, ведь они не роботы, чтобы работать без ошибок, хотя и роботы тоже иногда делают ошибки.

На первый взгляд все верно — доступ запрещен для всех, а открыт только для одного IP-адреса. Проблема кроется в том, что если посылка будет с адреса 192.168.1.1, то проверка первого правила для параметров пакета даст положительный результат, поэтому будет произведен запрет, пакет удалится, и второе правило не сработает никогда.

Чтобы наша политика действовала верно, строки надо поменять местами. При этом сначала будет проверена запись «Разрешить пакеты на 21-й порт с компьютера с адресом 192.168.1.1» — контроль пройдет успешно, а значит, пакет будет пропущен. Для остальных IP-адресов это правило не подойдет, и проверка продолжится. И вот тогда сработает запрет доступа на 21-й порт для всех пакетов.

Пакеты, направленные на другие порты, не будут соответствовать правилам, значит, над ними будут выполняться действия по умолчанию.

4.10.3. Брандмауэр — не панацея

Брандмауэр (Firewall) — это всего лишь замок на двери парадного входа. Злоумышленник редко пользуется парадным входом, он будет проникать в систему через черный или полезет в окно.

Сетевые экраны могут работать на компьютере с ОС (программные) или на каком-нибудь физическом устройстве (аппаратные). В любом случае — это программа, а ее пишут люди, которым свойственно ошибаться. Как и ОС, так и программу сетевого экрана нужно регулярно обновлять и исправлять погрешности.

Рассмотрим защиту по портам. Допустим, что у вас есть веб-сервер, который защищен сетевым экраном и в нем разрешен только порт 80. А хакеру больше и не надо!!! Ведь это не значит, что нельзя будет использовать другие протоколы. Можно создать туннель, через который данные одного протокола передаются внутри другого. Так появилась знаменитая атака Loki, которая санкционирует передачу команды для выполнения на сервере через ICMP-сообщения Echo Request (эхо-запрос) и Echo Reply (эхо-ответ), подобно команде ping.

Ну и еще один аргумент — наличие сетевого экрана ничего еще не означает, если его не настроить.

4.10.4. Брандмауэр как панацея

Может сложиться впечатление, что брандмауэр — это пустое развлечение и трата денег. Это не так. Если он хорошо настроен и постоянно контролируется, а в системе используются защищенные пароли, то сетевой экран может предотвратить большинство проблем.

Хороший экран имеет множество уровней проверки прав доступа, и нельзя использовать только один из них. Если вы ограничиваете доступ к Интернету исключительно по IP-адресу, то приготовьтесь оплачивать большой трафик. Но если при проверке прав доступа используется IP-адрес в сочетании с MAC-адресом и паролем, то такую систему взломать уже намного сложнее.

Ничего не понятно? Страшно, аж жуть? Если вы впервые видите команды управления сетевым экраном Linux, то только такие чувства могут вызывать на первых порах параметры утилиты `iptables`. Но ничего, хороший мануал и немного практики, и вы не будете бояться этих правил. Я надеюсь, что моя книга станет для вас хорошим мануалом, а практика придет со временем.

Давайте сделаем паузу, скушаем «твикс» и пока не будем рассматривать команды, а погрузимся в теорию и изучим, какие параметры может принимать команда `iptables`.

4.10.6. Основные возможности *iptables*

Вот список основных параметров, которые можно передавать команде `iptables`:

- `-A` цепочка правило — добавить правило в конец цепочки. В качестве параметров указывается имя цепочки (`INPUT`, `OUTPUT` или `FORWARD`) и правило;
- `-D` цепочка номер — удалить правило с указанным номером из заданной цепочки;
- `-R` цепочка номер правило — заменить правило с указанным номером в цепочке;
- `-I` цепочка номер правило — вставить правило в указанную первым аргументом цепочку, под номером, заданным во втором параметре. Если номер равен 1, то правило станет первым в цепочке;
- `-L` цепочка — просмотреть содержимое указанной цепочки;
- `-F` цепочка — удалить все правила из цепочки;
- `-p` протокол — определяет протокол, на который воздействует правило;
- `-i` интерфейс — определяет сетевой интерфейс, с которого данные были получены. Этот параметр можно использовать только с цепочками `INPUT`, `FORWARD` или `PREROUTING`;
- `-o` интерфейс — задает интерфейс, на который направляется пакет. Этот параметр можно использовать только с цепочками `OUTPUT`, `FORWARD` или `POSTROUTING`;
- `-j` действие — операция, которая должна быть выполнена над пакетом. В качестве аргументов можно указать следующие значения (рассмотрим только основные):
 - `LOG` — поместить в журнал запись о получении пакета;
 - `REJECT` — отправителю будет направлено сообщение об ошибке;
 - `DROP` — удалить пакет без информирования отправителя;
 - `BLOCK` — блокировать пакеты;
- `-s` адрес — IP-адрес отправителя пакета. После адреса можно задавать маску через косую черту (/), а перед адресом — знак отрицания !, что будет соответствовать любым адресам, кроме указанных;
- `-d` адрес — адрес назначения пакета.

выйдет из строя. Соответственно, хакер может просто направить бесконечные обращения на запрещенный порт и через некоторое время добиться удачно завершенной DoS-атаки.

Чтобы лучше понять работу фильтрации, желательно увидеть как можно больше примеров, поэтому дальше мы будем много практиковаться. Для начала посмотрим классический пример указания прав доступа, необходимых любому веб-серверу, т. е. разрешение обращений к 80-му порту нашего компьютера и возврат данных от нашего компьютера:

```
iptables -A INPUT -p tcp -m tcp --dport 80 -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp --sport 80 -j ACCEPT
```

В первой строке мы добавляем новое правило в цепочку `INPUT`. Правило действует на протокол TCP, о чем говорит значение `tcp` ключа `-m`. Веб-сервер использует как раз этот протокол для передачи данных. С помощью ключа `--dport` указываем порт назначения данных: 80. И наконец, последний параметр `-j ACCEPT` указывает, что если пакет соответствует всем правилам, то нужно его пропустить и обработать.

Сложно? Попробую другими словами: первая строка разрешает любые входящие пакеты по протоколу TCP, в которых происходит подключение на порт 80. Вторая строка разрешает передавать ответы от сервера.

Для того чтобы разрешить подключения по HTTPS, нужно добавить следующие две строки:

```
iptables -A INPUT -p tcp -m tcp --dport 443 -j ACCEPT
iptables -A OUTPUT -p tcp -m tcp --sport 443 -j ACCEPT
```

Здесь делается то же самое, только разрешается работа с портом 443.

Следующий пример разрешает подключения по SSH:

```
iptables -A INPUT -p tcp --dport 22 -j ACCEPT
iptables -A OUTPUT -p udp --sport 22 -j ACCEPT
```

Следующая команда создает фильтр, по которому запрещается принимать Echo-запросы от любых компьютеров:

```
iptables -A INPUT -s 0/0 -d localhost \
-p icmp --icmp-type echo-request -j DROP
```

Здесь параметр `-s 0/0` определяет, что правило относится ко всем адресам отправителя. Как и всегда в таких случаях, символ `\` указывает на то, что следующая строка является продолжением предыдущей. Вы можете убрать этот символ и записать все в одну строку.

Следующая команда запрещает доступ к FTP-порту:

```
iptables -A INPUT -s 0/0 -d localhost -p tcp --dport 21 -j DROP
```

Чтобы запретить доступ с определенного интерфейса, добавим ключ `-i` и укажем интерфейс `eth0`:

```
iptables -A INPUT -i eth0 -s 0/0 -d localhost -p tcp --dport 21 -j DROP
```


Кроме того, можно использовать NAT (Network Address Translation, трансляция сетевых адресов). Для чего нужна трансляция? Сетевых IP-адресов на всех не хватает, поэтому локальные сети используют адреса из зарезервированного диапазона (10.x.x.x или 192.168.x.x). Эти адреса не маршрутизируются и не могут использоваться в Интернете. Но как же тогда использовать в таких сетях Интернет? Все пакеты, направляемые на адреса в Интернете, перенаправляются на шлюз, на котором работает NAT. С помощью NAT адрес компьютера из локальной сети подменяется интернет-адресом, и пакет продолжает свой путь из локальной сети во Всемирную. Таким образом, достаточно иметь только один интернет-адрес, который присваивается шлюзу, и вся локальная сеть сможет работать с ресурсами Всемирной сети.

Для настройки NAT команда может выглядеть следующим образом:

```
iptables -t nat -A FORWARD -o ppp0 -j MASQUERADE
```

Ключ `-t nat` указывает на необходимость загрузить модуль `iptable_nat`. Если он не загружен, то это легко сделать вручную с помощью следующей команды:

```
modprobe iptable_nat
```

Здесь `iptable_nat` — модуль ядра, который позволяет сетевому экрану работать с NAT.

4.10.8. Утилита `firewalld`

Чуть более простой утилитой для управления сетевым экраном является `firewalld`, которая используется в CentOS и Red Hat. Она работает поверх `iptables` и предоставляет дополнительные функции сетевого экрана.

В Ubuntu по умолчанию `firewalld` не устанавливается, поэтому для ее установки выполняем следующую команду:

```
apt install firewalld
```

На главной странице проекта по адресу: <https://www.firewalld.org> вы можете узнать, что пакет `firewalld` отлично подходит для построения доверенных сетей и шлюзов в Интернет. Это тема отдельного разговора, и мы разовьем ее в *разд. 9.7*.

4.10.9. Uncomplicated Firewall: упрощенное управление

Столь сложное и интересное управление сетевым экраном, как представленное в предыдущих разделах, может пригодиться только сетевым администраторам. Домашние пользователи не захотят знать ничего про порты и протоколы, у них может быть одна задача — разрешить доступ программе, и они хотят ее выполнить как можно проще. И Ubuntu предоставляет им такую возможность с помощью уже упомянутой в *разд. «Фильтрация в Linux»* утилиты Uncomplicated Firewall (UFW).

Допустим, что вам нужно разрешить программе SSH подключаться к вашему серверу. Для этого просто выполняем команду:

```
ufw allow ssh
```


4.11.1. Обход сетевого экрана

Сетевой экран не может обеспечить абсолютной безопасности, потому что алгоритм его работы несовершенен. В нашем мире нет ничего безупречного, стопроцентно надежного, иначе жизнь стала бы скучной и неинтересной.

Как брандмауэр защищает ваш компьютер или сервер? Все базируется на определенных правилах, по которым экран проверяет весь проходящий через сетевой интерфейс трафик и выносит решение о возможности его пропуска. Но не существует такого фильтра, кроме абсолютного запрета, который может обеспечить полную безопасность.

На большинстве сетевых экранов возможно реализовать атаку DoS. Эта атака легко организуется в двух случаях:

- ☐ мощность канала злоумышленника больше, чем у вас;
- ☐ на сервере есть задача, требующая больших ресурсов компьютера, и есть возможность ее выполнить.

Сетевой экран — это сложная программная система, которой необходим значительный технический потенциал для анализа всего проходящего трафика, большая часть которого тратится на пакеты с установленным флагом `syn`, т. е. на запрос соединения. Параметры каждого такого пакета должны сравниваться со всеми установленными правилами.

В то же время для отправки `syn`-пакетов больших ресурсов и мощного канала не надо. Хакер без проблем может забросать разрешенный порт сервера `syn`-пакетами, в которых адрес отправителя подставляется случайным образом. Процессоры атакуемой машины могут не справиться с большим потоком запросов, которые надо сверять с фильтрами, и выстроится очередь, которая не позволит обрабатывать подключения добропорядочных пользователей.

Очень часто для совершения атаки нужно намного меньше ресурсов, чем для защиты. Если атака идет только с одного хоста, то его IP можно блокировать, но что, если атака идет сразу с многих IP-адресов (DDoS)? В таких случаях иногда блокируют целые сети. Например, есть такая компания Prolexic, которая защищает от DDoS-атак. Их мониторы следят за активностью и, судя по моему опыту работы с их системой, в случае атаки с одного адреса они его блокируют, а если с нескольких адресов одной сети, то они могут отключить всю сеть.

У меня один из сайтов клиента защищается этой компанией. Весь его трафик идет через ее центры данных, где интернет-каналы очень широкие и оборудование мощное. Но иногда случается так, что все пользователи крупнейшего провайдера Канады в Торонто не могут соединиться с каким-либо сайтом. Начинаем выяснять, а оказывается, что Prolexic просто заблокировал всю сеть.

Атака не может длиться вечно, и блокирование целыми сетями приносит свои плоды. Тем более что этот сайт — для американцев, канадцам там пользоваться нечем, поэтому чуть что, и их отключают.

Если клиент посылает слишком много данных, которые не могут быть помещены в один пакет, то информация разбивается на несколько блоков. Этот процесс назы-

4.11.2. Безопасный Интернет

Интернет не будет безопасным, пока нельзя четко установить принадлежность пакета. Любое поле IP-пакета можно подделать, и сервер никогда не сможет определить подлинность данных. Мы не в состоянии гарантировать спокойную жизнь, пока не можем четко знать источник и конкретного человека. С другой стороны, изменение ситуации может повлиять на нашу свободу.

В особо важных сетях должны пресекаться любые разведывательные действия — например, использование сканеров портов, трассировка сети и др., чтобы злоумышленник знал о сети как можно меньше.

Что такое трассировка? В сети каждый пакет проходит по определенному пути. При необходимости перенаправления такой пакет обязательно проходит через маршрутизаторы, которые доставляют его в нужные сети. Но если в устройство, обеспечивающее межсетевую совместимость, закралась ошибка, то пакет может навечно заблудиться. Чтобы этого не произошло, в заголовке IP-пакета есть поле TTL (Time To Live, время жизни). Отправитель пакета устанавливает в это поле определенное число, а каждый маршрутизатор уменьшает счетчик ретрансляций. Если значение TTL становится равным нулю, то пакет считается потерявшимся и уничтожается, а отправителю посылается сообщение о недостижимости хоста.

Эту особенность стали использовать для диагностики сети, чтобы узнать маршрут, по которому проходит пакет. Как это работает? В большинстве случаев каждый запрос идет до адресата одним и тем же путем. На пакет, отправленный со значением TTL, равным 1, первый же маршрутизатор ответит ошибкой, и по полученному отклику можно узнать его адрес. Следующая посылка идет с TTL, равным 2. В ответ на это ошибку вернет второй маршрутизатор. Таким образом можно узнать, через какие узлы проходят запросы к адресату.

Сетевой экран должен уничтожать любые пакеты с TTL, равным 1. Это защищает сеть, но явно указывает на наличие брандмауэра. Пакет с реальным значением TTL дойдет до адресата, а если команда `tracert` выдала ошибку, то это значит, что на пути следования пакета есть брандмауэр, который запрещает трассировку.

Для выполнения трассировки в ОС Linux нужно выполнить команду `tracert` с ключом `-I`, указав имя хоста. Например:

```
tracert -I redhat.com
```

В ОС Windows есть аналогичная команда `tracert`, и в ней достаточно задать имя узла или IP-адрес, который нужно трассировать, без использования дополнительных ключей.

При выполнении команды на экране начнут появляться адреса промежуточных маршрутизаторов, через которые проходит пакет. Например, результат может быть следующим:

```
tracert to redhat.com (xxx.xxx.xxx.xxx)? 30 hops max, 38 byte packets
 1  218 ms 501 ms 219 ms  RDN11-f200.101.transtelecom.net [217.150.37.34]
 2  312 ms 259 ms 259 ms  sl-gw10-sto-5-2.sprintlink.net [80.77.97.93]
 ...
 ...
```

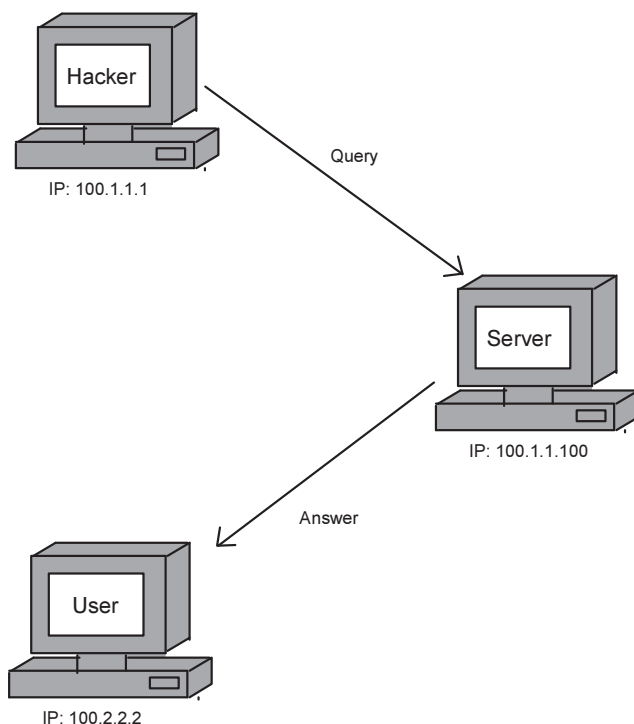



Рис. 4.3. Подмена IP-адреса

торой сервер найдет реальный адрес хакера 100.1.1.1. Поэтому в случае с TCP не так просто подделать адрес.

Современные сетевые экраны (в том числе и поставляемые с Linux) легко определяют подделку и блокируют такие пакеты.

4.12. Запрет и разрешение хостов

Работа с `iptables` может показаться сложной, потому что требует знания необходимых портов, но этот способ наиболее надежный, и для построения реальной защиты рекомендуется использовать именно его. А вот для решения простых задач есть другой метод — использование файлов `/etc/hosts.allow` и `/etc/hosts.deny`. Первый файл содержит записи хостов, которым разрешен доступ в систему, а во втором прописаны запреты.

При подключении к серверу файлы проверяются следующим образом:

1. Если соответствие обнаружено в файле `hosts.allow`, то доступ разрешен, и файл `hosts.deny` не проверяется.
2. Если в файле `hosts.deny` найдена запись, то доступ запрещен.
3. Если нет ни одной подходящей записи в обоих файлах, то доступ по умолчанию разрешен.

Но возможности этих файлов слишком ограничены, и любой сетевой экран позволяет осуществлять куда более тонкие настройки.

Я рекомендую использовать файлы `/etc/hosts.allow` и `/etc/hosts.deny` для решения временных проблем безопасности. Если в каком-либо сервисе найдена уязвимость, то его легко обойти через установки в файле `/etc/hosts.deny`. Если вы заметили попытку атаки с какого-нибудь IP-адреса, запретите на пару часов любые подключения с него, но, опять же, используя файл `/etc/hosts.deny`.

Почему нежелательно играть с цепочками сетевого экрана? Случайное удаление или добавление ошибочной записи может нарушить работу сервера или понизить его безопасность. Именно поэтому я не рекомендую устанавливать в сетевом экране временные правила.

4.13. Советы по конфигурированию брандмауэра

Конфигурирование сетевого экрана достаточно индивидуально и зависит от конкретных задач, решаемых сервером. Но я все же дам ряд рекомендаций, которым надо следовать во время настройки:

- изначально необходимо все запретить. К хорошему быстро привыкаешь, и если открыть что-то лишнее, то потом отучить пользователей будет трудно, и процесс закрытия сервиса станет проходить с большими сложностями;
- если есть возможность, необходимо запретить все типы ICMP-сообщений. Мы еще не раз будем говорить об опасности сканирования сети с помощью ICMP-пакетов;
- запретить доступ к порту 111. На нем работает `portmapper`, который необходим для удаленного вызова процедур (RPC, Remote Procedure Call) на сервере и получения результата. С помощью утилиты `rpcinfo` хакер может узнать, какие RPC-сервисы работают на вашем сервере. Например, выполните следующую команду:

```
rpcinfo -p localhost
```

Результатом будет примерно следующее:

```
Program vers proto  port
100000    2    tcp    111  portmapper
100000    2    udp    111  portmapper
100024    1    udp    32768 status
100024    1    tcp    32768 status
391002    2    tcp    32769 sgi_fam
```

Как видите, одна команда может сообщить весьма большое количество информации, поэтому 111-й порт необходимо закрыть;

- для облегчения управления доступом к портам разделите открытые ресурсы на две категории:

образом мы сможем обезопасить себя от возможных проблем. Пример содержимого этого конфигурационного файла приведен в листинге 4.2.

Листинг 4.2. Содержимое конфигурационного файла /etc/sudoers

```
# User privilege specification
root    ALL=(ALL)  ALL

# Uncomment to allow people in group wheel to run all commands
# %wheel    ALL=(ALL)    ALL

# Same thing without a password
# %wheel    ALL=(ALL)    NOPASSWD: ALL

# Samples
# %users    ALL=/sbin/mount /cdrom,/sbin/umount /cdrom
# %users    localhost=/sbin/shutdown -h now
```

В этом файле только одна строка без комментария:

```
root    ALL=(ALL)  ALL
```

Она состоит из трех параметров:

- ☐ имя — пользователь (или группа), которому разрешено выполнять определенную команду. Я рекомендую указывать конкретных пользователей. Хакер может стать участником группы и, не обладая при этом частными привилегиями, получает доступ к выполнению опасных команд;
- ☐ компьютер — имя машины, на которой можно выполнять команду от лица администратора;
- ☐ команды, которые разрешено выполнять указанному пользователю (указываются после знака «равно»).

Итак, чтобы пользователь смог просмотреть файл /etc/shadow, необходимо прописать соответствующее право. В моей системе есть простой пользователь с именем robert. Для него я добавляю в файл /etc/sudoers следующую запись:

```
robert    ALL=ALL
```

Теперь пользователь robert сможет выполнять с помощью `sudo` любые администраторские задачи. Проверьте это, повторив выполнение команды:

```
sudo cat /etc/shadow
```

На этот раз все должно пройти успешно — на экране появится приглашение ввести пароль администратора для выполнения команды.

Но разрешение выполнения абсолютно всех команд не соответствует принципам построения безопасной системы. Необходимо вводить определенные ограничения.

Обслуживать сервер, который обрабатывает ежедневно множество подключений пользователей и на котором работают разные сервисы, в одиночку очень сложно.

ватель сможет зарегистрироваться в системе как администратор и сделать все что угодно.

Желательно не передавать никому пароль администратора. Используйте пароли других учетных записей, которым разрешена работа с необходимыми файлами и программами. В этом случае придется указывать конкретное имя пользователя, которое назначил администратор для выполнения команды.

Еще один способ сохранить пароль администратора — разрешить пользователю выполнять команды без аутентификации. Для этого необходимо между знаком равенства и списком разрешенных команд добавить ключевое слово `NOPASSWD` и двоеточие. Например:

```
robert ALL=NOPASSWD:/bin/cat /etc/shadow, /bin/mount /dev/cdrom
```

Теперь при выполнении команды `sudo` пароль вообще запрашиваться не будет. Это не совсем безопасно, если вы не указываете необходимые команды, а лишь ключевое слово `ALL`:

```
robert ALL=NOPASSWD:ALL
```

И если хакер получит доступ к учетной записи `robert`, то сможет с помощью команды `sudo` выполнять в системе любые команды. Если же вы указываете возможные директивы, то опасность взлома системы уменьшается в зависимости от того, насколько опасные команды вы разрешаете выполнять пользователю `robert` и в какой мере защищена эта учетная запись (длина и сложность пароля, прилежность владельца и т. д.).

С помощью утилиты `sudo` можно предоставить доступ для корректировки файлов, но я никогда не делегирую возможность корректировки конфигурационных файлов с помощью редакторов.

К потенциально опасным командам, которые нежелательно предоставлять для выполнения с правами `root` другим пользователям, относятся:

- ☐ редактирование файлов — позволяет злоумышленнику изменить любой конфигурационный файл, а не тот, что вы задали;
- ☐ `chmod` — дает возможность хакеру понизить права доступа на конфигурационный файл и после этого редактировать его даже с правами гостя;
- ☐ `useradd` — добавляет учетные записи. Если хакер создаст пользователя с ID, равным нулю, то он получит полные права в системе;
- ☐ `mount` — позволяет монтировать устройства. Прописывайте в конфигурационном файле конкретные устройства и доверяйте эту команду только проверенным пользователям. Если хакер смонтирует устройство со своими программами, которые будут содержать SUID-биты, или троянскими программами, то он сможет получить в свое распоряжение всю систему;
- ☐ `chgrp` и `chown` — санкционируют смену владельца файла или группы. Изменив владельца файла паролей на себя, хакер сможет легко его прочитать или даже изменить.


```
When to require a password for 'list' pseudocommand: any
When to require a password for 'verify' pseudocommand: all
Local IP address and netmask pairs:
    192.168.77.1 / 0xffffffff00
Default table of environment variables to clear
    BASH_ENV
    ENV
    TERMCAP
    ...
    ...
Default table of environment variables to sanity check
    LANGUAGE
    LANG
    LC_*
```

Я не стал приводить результат выполнения команды полностью, но основную информацию здесь можно увидеть. Вначале нам сообщается версия программы `sudo` — в нашем случае это 1.6.5.p2. Наиболее интересными в этом листинге являются следующие три строки:

```
Authentication timestamp timeout: 5 minutes
Password prompt timeout: 5 minutes
Number of tries to enter a password: 3
```

В первой строке указывается время сохранения пароля в кэше. В нашем случае это 5 минут. Если пользователь в течение этого времени снова выполнит команду `sudo`, то повторно аутентификация проводиться не будет.

Следующая строка указывает время ожидания ввода пароля, а последняя — количество попыток его задать. Если за этот период верный пароль не будет указан, работа программы прервется.



ГЛАВА 5

Администрирование

В этой главе мы рассмотрим вопросы, с которыми администраторы Linux-систем ежедневно сталкиваются в своей работе. Нам предстоит познакомиться со множеством команд Linux, научиться их использовать и узнать немало полезного о системе.

Но изучением команд мы не ограничимся, потому что иначе книга превратится в перевод руководства по Linux. Чтобы этого не случилось, я постарался подобрать готовые решения задач, которые мне самому приходится выполнять в повседневной жизни. Я понимаю, что все объять в одной книге не получится, но надеюсь, что материалы этой главы помогут вам найти ответы на многие вопросы и избавиться хотя бы от части проблем.

5.1. Полезные команды для сетевых соединений

Для начала познакомимся с некоторыми программами и командами, которые позволят вам упростить администрирование и сделать его эффективнее. И начнем мы с команд, необходимых для понимания последующего материала.

Я только однажды работал чисто системным администратором, а все остальное время занимаюсь программированием, и очень часто мне приходится поддерживать и сайты. Когда я занимался разработкой сайтов Sony, то в случае проблем все бежали ко мне, а не к администраторам, потому что я могу диагностировать проблему не только с точки зрения администратора, но и программиста.

И когда что-то происходит с сервером, то очень часто приходится проверять первым делом сетевое соединение: нет ли проблем связи с серверами приложений, доступна ли база данных и т. д. Команды работы с сетью, наверное, самые популярные.

5.1.1. *ping*

Одна из часто используемых администраторами команд — `ping`. Она посылает ICMP-пакеты в виде эхо-запросов на указанную систему для определения пропускной способности сети. О таких запросах мы уже говорили, когда рассматривали сетевой экран (см. *разд. 4.10*).

Например, выполните команду: `ping 195.18.1.41`, и в ответ вы получите информацию примерно такого вида:

```
PING 195.18.1.41 (195.18.1.41) from 195.18.1.41 : 56(84) bytes of data.  
64 bytes from 195.18.1.41: icmp_seq=1 ttl=64 time=0.102 ms  
64 bytes from 195.18.1.41: icmp_seq=2 ttl=64 time=0.094 ms  
64 bytes from 195.18.1.41: icmp_seq=3 ttl=64 time=0.094 ms  
64 bytes from 195.18.1.41: icmp_seq=4 ttl=64 time=0.095 ms  
--- 195.18.1.41 ping statistics ---  
4 packets transmitted, 4 received, 0% loss, time 3013ms  
rtt min/avg/max/mdev = 0.094/0.096/0.102/0.007 ms
```

В этой команде IP-адрес можно заменить любым адресом в вашей сети или в Интернете, например:

```
ping microsoft.com
```

Первая строка — информационная, в ней отображается IP-адрес компьютера, с которым будет происходить обмен сообщениями (если вы указали символическое имя хоста, то простой командой `ping` можно узнать его IP-адрес). В конце строки указан размер пакетов данных, которые будут отправляться.

Следом на экране начнут появляться строки вида:

```
64 bytes from 195.18.1.41: icmp_seq=1 ttl=64 time=0.102 ms
```

Отсюда мы узнаем, что с адреса 195.18.1.41 было получено 64 байта. После двоеточия отображаются следующие параметры:

- ❑ `icmp_seq` — номер пакета. Это значение должно последовательно увеличиваться на единицу. Если какой-либо номер отсутствует, то это означает, что пакет потерялся в Интернете и не дошел до адресата или ответ не вернулся к вам. Это может происходить из-за неустойчивой работы оборудования, плохого кабельного соединения или в случае, когда один из маршрутизаторов в сети между компьютерами выбрал неправильный путь и пакет не достиг места назначения;
- ❑ `ttl` — время жизни пакета. При отправке пакета ему отводится определенное время жизни, которое задается целым числом. По умолчанию в большинстве версий `ttl` равен 64, но это значение может быть изменено. Каждый маршрутизатор при передаче пакета уменьшает значение `ttl` на единицу. Когда оно становится равным нулю, пакет считается заблудившимся и уничтожается. Таким образом, по этому значению можно приблизительно сказать, сколько маршрутизаторов встретилось на пути;
- ❑ `time` — время, затраченное на ожидание ответа. По этому параметру можно судить о скорости связи и ее стабильности, если значение параметра не сильно

изменяется от пакета к пакету. Но учитывайте, что время, затраченное на отправку и получение первого пакета, почти всегда выше, особенно при указании имени, а не IP. Это связано с затратами на поиск в базе DNS. Все остальные пакеты должны идти равномерно.

Если ответ на какой-нибудь запрос не был получен, то вы увидите сообщение о его потере. Дождитесь, пока программа не отправит 7–10 пакетов, чтобы по их параметрам оценить качество связи, после чего можно прервать сеанс нажатием комбинации клавиш <Ctrl>+<C>. В результате вы увидите краткую статистику обмена сообщениями: количество отправленных, полученных и потерянных пакетов, а также минимальное, среднее и максимальное время обмена пакетами.

Вот основные параметры, которые можно использовать в команде `ping`:

- ❑ `-c n` — завершение работы после отправки (и приема) `n` пакетов. Например, вы хотите послать пять запросов, тогда команда будет выглядеть следующим образом:

```
ping -c 5 195.10.14.18
```

- ❑ `-f` — форсированная передача. Например, вам нужно быстро отправить 50 запросов, тогда команда будет выглядеть так:

```
ping -f -c 50 195.10.14.18
```

Имейте при этом в виду, что ключ `-f` в сочетании с большим количеством пакетов значительного размера может сильно загрузить сеть и компьютер получателя, а в некоторых системах даже привести к временному отказу от обслуживания;

- ❑ `-s n` — размер пакета. Например, если вы хотите отправить пакет размером в 1000 байтов, команда должна выглядеть так:

```
ping -s 1000 195.10.14.18
```

В некоторых старых версиях ОС были ошибки, и при получении слишком большого пакета система зависала. В настоящее время погрешности такого рода отсутствуют, и встретить подобную систему в Интернете сложно.

Это наиболее часто используемые параметры. Дополнительную информацию по команде можно получить в справочной системе, выполнив команду: `man ping`.

ВНИМАНИЕ!

Не каждый сервер отвечает на эхо-запросы. В некоторых системах сетевой экран может быть настроен так, чтобы не пропускать ICMP-трафик, и тогда ответа не будет, хотя реально сервер работает в нормальном режиме и пакеты другого типа может воспринимать без проблем.

5.1.2. *netstat*

Эта команда показывает текущие подключения к компьютеру. Результат ее выполнения имеет примерно следующий вид:

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address      Foreign Address    State
```

```

tcp      0      0 FlenovM:ftp      192.168.77.10:3962 ESTABLISHED
tcp      0      0 FlenovM:ftp-data 192.168.77.10:3964 TIME_WAIT

```

Информация представлена в виде таблицы. Давайте рассмотрим ее колонки:

- ❑ Proto — базовый протокол, который использовался для соединения. Чаще всего здесь можно видеть `unix` или `tcp`;
- ❑ Recv-Q — количество байтов, не скопированных пользовательской программой из очереди;
- ❑ Send-Q — количество байтов в очереди на отправку удаленному компьютеру;
- ❑ Local Address — локальный адрес формата `компьютер:порт`. В качестве порта может использоваться как символьное имя, так и число. В приведенном примере в первой строке показан порт `ftp`, что соответствует числу 21;
- ❑ Foreign Address — удаленный адрес в формате `IP:порт`;
- ❑ State — состояние соединения.

У этой команды множество дополнительных параметров, и полное их описание можно увидеть в файле справки, набрав команду: `man netstat`.

В случае непредвиденной ситуации и подозрения на проникновение в систему извне вы с помощью этой команды сможете определить, через какой сервис произошло вторжение и что хакер в текущий момент может использовать. Например, если злоумышленник пробрался через FTP, то он, скорее всего, работает с файлами и может закачивать свои программы для дальнейшего взлома или удалять системные файлы (это зависит от прав доступа).

5.1.3. *telnet*

Мощность Linux и ее текстовой консоли заключается в том, что вы можете выполнять все действия не только сидя непосредственно за терминалом, но и удаленно, за тысячи километров. Нужно лишь подключиться к компьютеру на порт Telnet-сервера с помощью Telnet-клиента, и можно считать, что вы в системе и к вашим услугам мощности производительной техники, которая может быть недоступна домашнему пользователю.

В Windows очень мало утилит, способных работать в командной строке, поэтому в этой ОС необходим и широко используется графический режим. Да и сама командная строка в Windows стала более или менее продвинутой совсем недавно (с появлением PowerShell). Для решения этой проблемы был создан терминальный доступ, который позволяет на клиентской машине видеть экран сервера и работать с ним так, как будто вы сидите непосредственно перед ним. Но этот метод отнимает слишком много трафика и очень неудобен на медленных каналах связи.

Командная строка Linux по сравнению с графическим режимом практически не расходует трафик и может приемлемо работать даже по самым медленным каналам — например, по соединениям GPRS сотового телефона или с домашнего модема, где скорость довольно мала.

Как вы уже поняли, программное обеспечение Telnet состоит из серверной и клиентской частей. При запуске Telnet-сервера открывается порт 23, к которому можно подключиться с клиентского компьютера и выполнять любые команды, которые позволяет удаленный сервер.

Но это еще не все — с помощью Telnet-клиента можно подсоединяться и к любому сервису, протокол которого является текстовым (т. е. команды читаемы и представляют собой текст). Например, можно подключиться на порт 25 и прямо из командной строки отправить E-mail-сообщение, подавая команды SMTP-сервера.

Если у вас есть установленный FTP-сервер, то уже сейчас вы можете выполнить команду:

```
telnet localhost 21
```

Первый параметр здесь — это localhost, потому что я подразумеваю, что вы подключаетесь к локальному FTP-серверу. Если вы хотите работать с удаленным компьютером, то укажите его адрес. В качестве второго параметра указывается номер порта. Сервер FTP принимает управляющие команды на порту 21, поэтому я указал это число.

Точно так же я проверяю и веб-сайты. Если сайт не загружается, то можно проверить, отвечает ли хотя бы веб-сервер, и для этого можно попытаться подключиться к 80-му порту:

```
telnet flenov.info 80
```

Если соединение установлено, то веб-сервер работает, и можно уже проверять дальше, что именно недоступно.

Я рекомендую применять Telnet-клиент только для отладки различных сервисов, но не для управления реальным компьютером. Поэтому на всех «боевых» серверах отключайте сервер Telnet. Он небезопасен, потому что данные при Telnet-управлении передаются в открытом виде и могут быть перехвачены. Все попытки сделать Telnet защищенным заканчивались неудачно и не приводили к желаемому результату. Один из относительно безопасных вариантов использования Telnet — это подключение через канал шифрования OpenSSL (от Secure Socket Layer, протокол защищенных сокетов). Но наибольшую популярность получило управление сервером через протокол SSH (OpenSSH, Open Secure Shell), который мы рассмотрим позже, в *разд. 5.3*.

Таким образом, Telnet-клиент в системе необходим, а Telnet-сервер, если он у вас установлен, следует срочно удалить и забыть как страшный сон.

Если вы все же решили использовать Telnet-сервер, то это необходимо делать только через протокол для безопасной связи по сети с применением открытых и секретных ключей (см. *разд. 5.2*). Тогда весь трафик будет шифроваться.

Если у вас установлен Telnet-сервер, то попробуйте сейчас подключиться к нему, используя команду: `telnet localhost`. Если сервер запущен, и подключение разрешено сетевым экраном, то перед вами появятся сообщения примерно такого вида:

```
Trying 127.0.0.1
Connected to localhost
Escape character is '^['.
```

```
ASPLinux release 7.3 (Vostok)
Kernel 2.4.18-15asp on an i686
Login:
```

Очень плохо, что мы видим подробную информацию о дистрибутиве и ядре. И все это становится известным еще до регистрации любому посетителю. Если хакер увидит открытый порт 23, то ему уже не надо будет мучиться для выяснения, какая у вас ОС и версия ядра, — достаточно подключиться к Telnet, и проблема решена.

Излишняя болтливость Telnet — большая дыра, с которой надо бороться в первую очередь. Приветствие, которое вы можете видеть при подключении, находится в файлах: `/etc/issue` и `/etc/issue.net`. Измените текст сообщения, например, следующим образом:

```
echo Текст > /etc/issue
echo Текст > /etc/issue.net
```

Здесь Текст — это новое приветствие. Можно прописать и ложную версию ядра:

```
echo Debian Linux > /etc/issue
echo Kernel 2.4.4 on an i686 > /etc/issue.net
```

Не думаю, что так можно кого-то сильно запутать, и на реальную безопасность подобный обман не влияет, поэтому большинство специалистов просто рекомендуют не показывать лишнего о своей ОС.

Проблема в том, что после перезагрузки содержимое файлов восстановится, и Telnet в приветствии снова покажет всю информацию о системе. Чтобы этого не произошло, после изменения текста приветствия можно установить на файлы с приветствием атрибут `+i`, который запрещает любые изменения:

```
chattr +i /etc/issue
chattr +i /etc/issue.net
```

Но главная проблема Telnet — не болтливость, а отсутствие безопасности. Утилита передает все команды в открытом виде, без шифрования. Если хакеру удастся каким-либо образом перехватить ваши пакеты, то он сможет увидеть не только выполняемые вами команды, но и пароли доступа.

5.1.4. r-команды

В Linux есть так называемые r-команды: `rlogin`, `rsh`, `rcp`, `rsync`, `rdist`. Мы не будем их рассматривать, потому что все они создают большие проблемы в безопасности. Если Telnet-клиент полезен для тестирования сервисов, то эти команды я включил в обзор только для того, чтобы вы удалили их из системы, исключив тем самым соблазн их использовать и возможность их применения хакером.

Все r-команды устарели и небезопасны, потому что позволяют подключаться к системе удаленно и при этом передают данные без шифрования.

5.2. Шифрование

Во времена рождения Интернета и первых сетевых протоколов еще не задумывались о безопасности. О ней стали думать только тогда, когда начали происходить реальные взломы и целые вирусные эпидемии. Одним из самых больших упущений было то, что в большинстве протоколов данные по сети передаются в открытом виде, а сетевое оборудование позволяет прослушивать сетевой трафик.

В локальной сети есть свои особенности, потому что соединения могут осуществляться различными способами. При объединении компьютеров через центральное устройство — хаб (hub) или коммутатор (switch) — все устройства в сети подключаются к центральному устройству (рис. 5.1). В наше время все больше используется беспроводное соединение, но идея остается та же — по Wi-Fi множество компьютеров могут подключаться к одному центральному устройству (маршрутизатору).

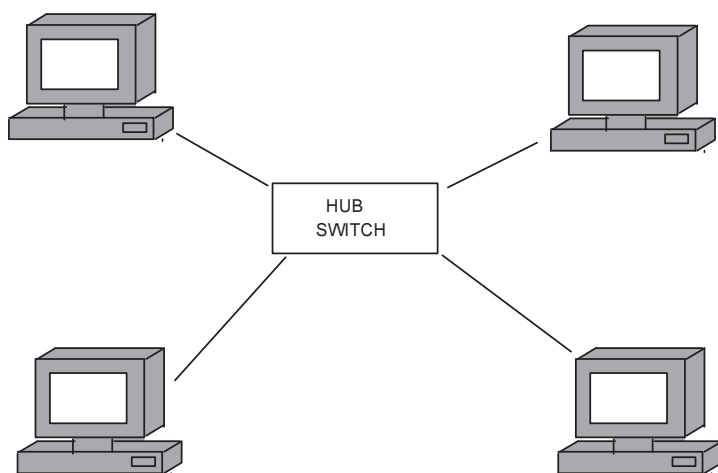


Рис. 5.1. Схема соединения компьютеров «звезда»

Если в центре сети стоит хаб (простое и дешевое устройство), то все пакеты, пришедшие с одного компьютера, копируются на все узлы, подключенные к этому хабу. Это устройство можно считать очень глупым, потому что оно не занимается маршрутизацией пакетов и не определяет устройство, которое должно быть получателем информации. Оно просто тупо отправляет всю информацию всем участникам сети, а они уже должны заниматься фильтрацией. Плюс таких устройств — они дешевле, но совершенно небезопасные.

Хабы сейчас выходят из моды, потому что процессоры становятся дешевле, и наделять сетевое оборудование мозгами уже не так дорого. Я оставил информацию о хабах только для того, чтобы вы могли знать о том, с какими проблемами приходилось раньше сталкиваться при работе с сетями.

Даже в домашних сетях сейчас в большинстве случаев используются умные коммутаторы, или маршрутизаторы (роутеры), которые поддерживают как проводное, так и беспроводное Wi-Fi соединение, а они уже умнее и направляют пакеты только получателям, а не всем подряд.

При работе через коммутаторы пакеты будет видеть только получатель, потому что это устройство имеет встроенные возможности маршрутизации, которые для локальных сетей могут реализовываться через MAC-адреса (Media Access Control Address, адрес управления доступом к среде), которые называют физическими адресами, или через IP-адреса.

Физический адрес — это 48-разрядный серийный номер сетевого адаптера, присваиваемый ему производителем. Он уникален, потому что у каждого изготовителя свой диапазон адресов (однако не стоит думать, что его невозможно изменить). К каждому порту коммутатора подключен компьютер с определенным MAC-адресом. Пакет направляется только на порт адресата, и другие участники сети его не увидят.

Существуют коммутаторы, которые умеют управлять передачей на уровне IP-адреса (логический адрес), как это делают маршрутизаторы (router). При этом пакеты отправляются исходя из логических, а не физических адресов, и коммутатор может объединять целые сети.

Но даже в случае использования коммутатора есть возможность прослушать трафик на сервере. Такое положение дел никого не устраивает, особенно при работе с конфиденциальными данными.

Переделывать существующие протоколы нереально, поскольку это накладно и в некоторых случаях просто невыполнимо, потому что потребует изменения всех существующих программ. С введением IPv6 разработчики учли некоторые недочеты старой версии IP, только вот 6-я его версия завоевывает популярность слишком медленно.

Уже найдено более удобное и универсальное решение — туннелирование (tunneling), которое позволяет программам удаленного доступа различных разработчиков взаимодействовать друг с другом, а также поддерживает несколько методов проверки подлинности, сжатия и шифрования данных. В общих чертах туннелирование выглядит следующим образом (на примере FTP):

- на клиентском компьютере на определенном порту (Порт 1) вы должны запустить программу для шифрования трафика. Теперь, вместо того чтобы передавать данные на удаленный компьютер, вам следует с помощью FTP-клиента соединиться с Портом 1 своего компьютера и направлять данные на него. Полученные данные будут шифроваться и передаваться по сети в закрытом программой шифрования виде;
- на удаленном компьютере на определенном порту запускается такая же программа — она принимает зашифрованные данные, дешифрует их и передает в открытом виде на порт FTP-сервера.

С помощью туннеля можно зашифровать протокол или работу программы, в которой шифрование изначально не реализовано. Для наиболее популярного протокола

HTTP уже давно реализовали версию с шифрованием — HTTPS, для которой не нужно открывать отдельный туннель.

На рис. 5.2 показано, как происходит шифрование данных, — все пакеты передаются через шифрующего их посредника. В настоящее время наиболее распространенным протоколом шифрования является SSL (Secure Sockets Layer, протокол защищенных сокетов). Он зарекомендовал себя как надежное средство обмена данными и уже многие годы защищает транзакции в Интернете.

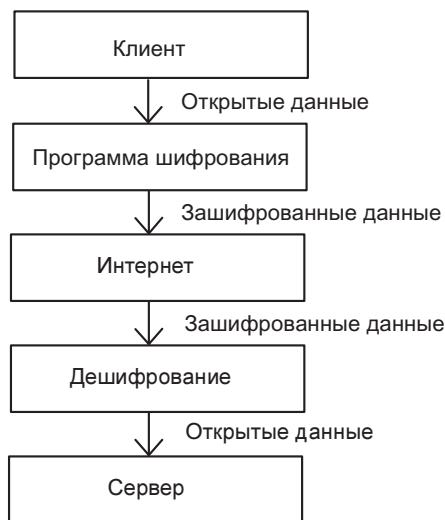


Рис. 5.2. Шифрование канала

Очень часто можно услышать сочетание SSL/TLS. Вторая часть — TLS — расшифровывается как Transport Layer Security. TLS-протокол основан на спецификации протокола SSL версии 3.0, который был выпущен в 1996 году. После этого он уже развивается TLS.

Были случаи, когда находили уязвимости и в TLS-протоколе, но их всегда очень быстро исправляли и оставалось только обновить софт.

Рассмотрим шифрование в сети. Когда вы покупаете в электронном магазине какой-либо товар, то в этот момент организуется безопасное соединение с шифрованием, чтобы ни один злоумышленник не смог подсмотреть информацию о вашей банковской карте. В момент подключения к серверу по HTTPS браузер автоматически запускает на компьютере клиента шифрование, в зашифрованном виде передает на сервер данные и в зашифрованном же виде получает ответы.

Выходит, что шифрование не изменяет протокол, он остается тем же самым TCP/IP, просто на сторонах клиента и сервера появляются посредники, которые шифруют и дешифруют данные. Использование такого метода очень удобно тем, что позволяет шифровать трафик любого протокола. И даже если в криптографическом алгоритме будет найдена ошибка или в него будут внесены изменения для более качественного шифрования (например, с использованием более длинного

ключа), то не придется вносить изменения в протокол. Достаточно обновить программу шифрования, и все новые возможности станут доступными.

Рассмотрим пример с веб-сервисом, который работает на порту 80. На сервере можно запустить программу шифрования — например, на порту 1443, и все данные, которые будут приходить на него, станут дешифровываться и передаваться на порт 80. Если вы хотите предоставлять доступ к сети только по протоколу SSL, то к порту 80 можно закрыть доступ извне при помощи сетевого экрана (о работе сетевого экрана мы подробно говорили в *главе 4*), а разрешить лишь подключения с сервера шифрования.

Адреса сайтов, которые защищены специальными ключами, начинаются с префикса **https://**. Буква **s** как раз и говорит о том, что соединение безопасно. Помимо этого, при подключении через браузер с включенным SSL где-либо в его окне — в строке состояния или в адресной — появляется значок с изображением висячего замка. Например, в популярном среди пользователей Windows браузере Internet Explorer (рис. 5.3) этот замок расположен в конце адресной строки, а в популярном браузере Chrome он находится в ее начале. В последней версии браузера от Microsoft — Edge — он также перенесен в начало адресной строки.

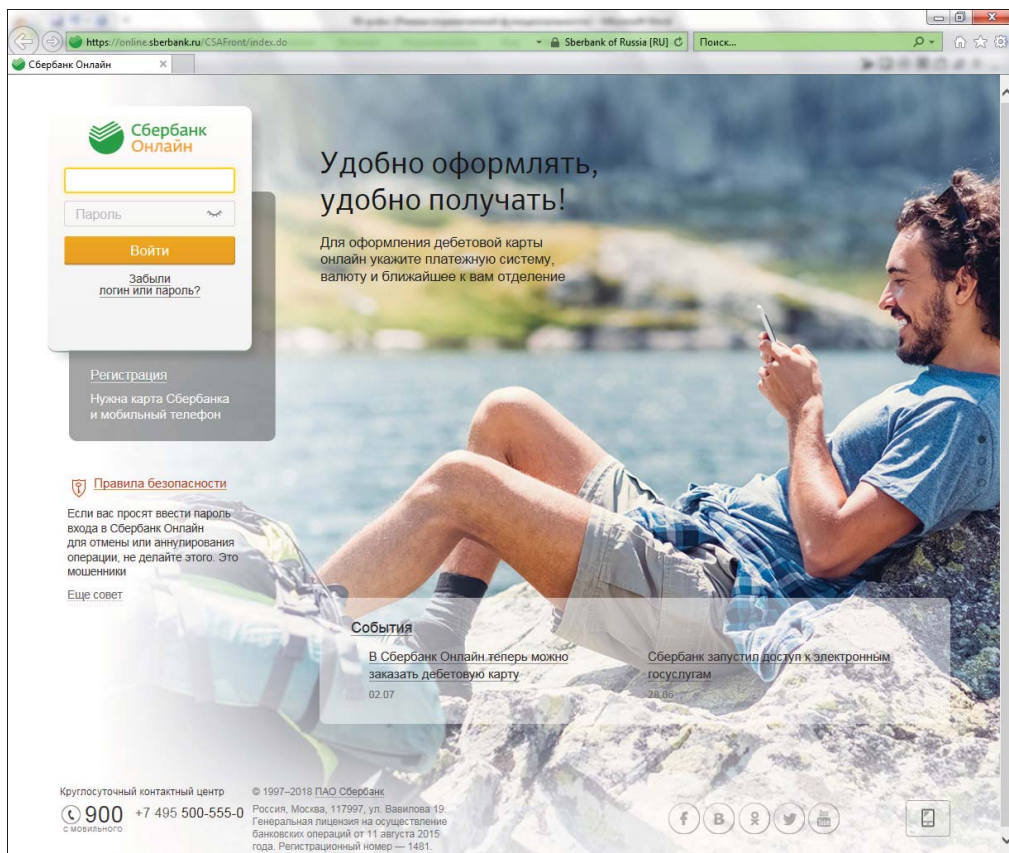


Рис. 5.3. Безопасное соединение в браузере Internet Explorer

Более точно определить тип используемого соединения можно по свойствам документа. Большинство браузеров имеют команду просмотра свойств загруженной страницы, вызвав которую можно увидеть и применяемое шифрование. Например, в окне Internet Explorer необходимо нажать правую кнопку мыши и выбрать из появившегося контекстного меню опцию **Свойства** (Properties).

Перед вами откроется окно, как на рис. 5.4. Обратите внимание на поле **Подключение** (Connection). Информация в нем свидетельствует, что используется протокол TLS 1.2, 256-разрядное шифрование AES и RSA с 2048-разрядным обменом.

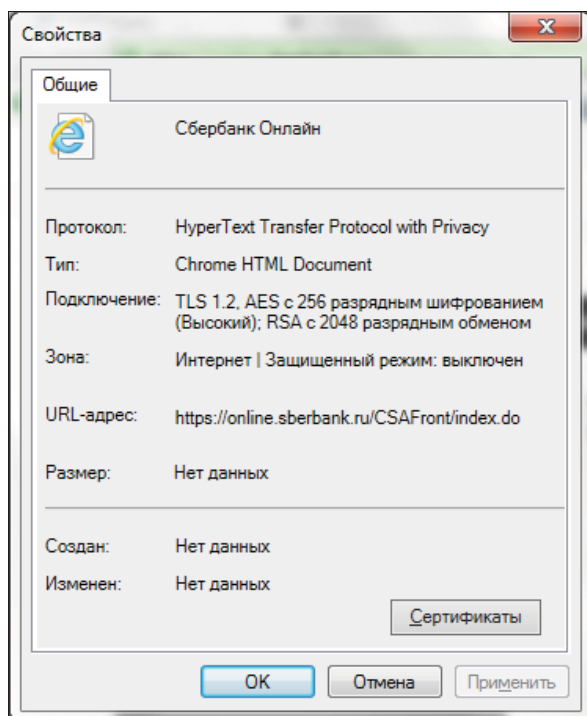


Рис. 5.4. Свойства документа в Internet Explorer

Для шифрования интернет-трафика нужен сертификат, который стоит денег. У таких сайтов, как Microsoft, Google, и других сайтов с высокой нагрузкой обработку запросов клиентов осуществляют весьма много серверов. Обычно серверы, которые обрабатывают запросы, находятся уже в защищенной сети, и к ним невозможно подключиться извне. Только определенный компьютер, который выполняет роль шлюза, может подключаться к ним. Он передает трафик извне и распределяет его по серверам. Эта достаточно простая схема, и она позволяет установить сертификат только на один-единственный сервер-шлюз. Он расшифровывает трафик и направляет его дальше уже не защищенным на определенный порт. А по номеру порта можно решить, является трафик защищенным или нет.

5.2.1. Программа stunnel

В ОС Linux для шифрования и дешифрования трафика чаще всего используется программа stunnel. Однако она лишь организует канал и выступает посредником, а для самого шифрования задействуется пакет OpenSSL, доступный в большинстве дистрибутивов Linux. Вероятно, он у вас уже установлен, а если нет, то его легко установить с помощью соответствующего пакета с установочного диска. Более подробную информацию по OpenSSL и его последние обновления можно найти на сайте: www.openssl.org.

В основу OpenSSL положено применение пары ключей: открытого и закрытого. С помощью открытого ключа можно только зашифровать данные, и он посылается клиентом на сервер по открытому каналу (отсюда и его название). Сервер использует его для шифрования, но для расшифровки необходим закрытый ключ, который известен только клиенту.

У программ OpenSSL и stunnel очень много параметров, и рассматривать все их здесь бессмысленно, потому что запомнить весь их набор проблематично, да и обычно не нужно. Лучше мы разберем реальный пример и познакомимся с наиболее часто используемыми параметрами этих программ.

Итак, давайте для начала запустим на сервере программу stunnel, которая будет расшифровывать входящий трафик и передавать его на какой-нибудь порт — например, 25 (здесь работает SMTP-сервер sendmail). Для этого выполните следующую команду:

```
stunnel -p /usr/share/ssl/cert.pem -d 9002 -r 25
```

Здесь используются три параметра:

- ❑ `-p` — ключ, после которого указывается SSL-сертификат авторизации, по умолчанию уже созданный во время установки ОС или программы stunnel и находящийся в файле `/usr/share/ssl/cert.pem`;
- ❑ `-d` — показывает, что туннель должен работать как домен. После этого ключа ставится IP-адрес (необязательный параметр) и порт, на котором нужно ожидать подключения. Если адрес не указан, как в нашем примере, то прослушиваться будут все интерфейсы локального компьютера. В качестве порта был выбран номер 9002. Все пришедшие на него данные считаются зашифрованными, и они будут дешифроваться для передачи на другой порт локального компьютера;
- ❑ `-r` — после этого ключа указывается имя компьютера (необязательный параметр) и порт, куда нужно передавать расшифрованные данные. Если хост не указан, то данные будут пересылаться на порт локального компьютера — в нашем случае на порт 25, где должен работать SMTP-сервер. Если данные предназначены другому компьютеру, то в качестве параметра укажите `192.169.77.1:25`, где `192.169.77.1` — это IP-адрес компьютера. Однако помните, что при этом незашифрованные данные будут пересылаться по сети, чего мы как раз пытались избежать.

Теперь рассмотрим запуск клиента. Тут все намного проще:

```
stunnel -c -d 1000 -r 192.168.77.1:9002
```

Здесь у нас появился еще один ключ: `-c` — он означает, что туннель создается для клиента (по умолчанию программа `stunnel` запускается в серверном режиме).

Помимо этого в примере запуска клиента присутствуют и уже известные параметры: `-d` — с указанием порта, на котором необходимо ждать подключения (значение 1000), и `-r`, в котором указывается адрес и порт для передачи зашифрованных данных.

Теперь достаточно настроить свою клиентскую программу так, чтобы при отправке почты она направлялась на порт (в нашем случае 1000) компьютера, где запущен `stunnel`-клиент, который будет шифровать данные и пересылать их на порт 9002 сервера с адресом 192.168.77.1. Там зашифрованные данные будет принимать `stunnel`-сервер, расшифровывать их и передавать на 25-й порт сервера.

Если на вашей системе включен сетевой экран и запрещены подключения по указанным портам, то описанные соединения сетевым экраном будут отклонены.

5.2.2. Дополнительные возможности OpenSSL

При запуске программы `stunnel` на сервере мы задали использование сертификата авторизации, но не сказали, как проверять его подлинность. Для указания уровня проверки подлинности сертификата авторизации служит ключ `-v`, после которого идет число:

- ☐ 0 — нет никакой проверки;
- ☐ 1 — если сертификат присутствует, то он проверяется на подлинность. Если получен отрицательный результат, то соединение разрывается. Наличие сертификата не является обязательным, соединение может быть установлено и без него;
- ☐ 2 — сертификат является обязательным. Если сертификата нет или он не является подлинным, соединение не может быть установлено;
- ☐ 3 — наличие сертификата обязательно, и помимо этого он должен присутствовать в локальном хранилище (специальный список). В этом случае с помощью опции `-a` нужно указать каталог, в котором находятся сертификаты.

Сервер SSL может расшифровывать трафик и передавать его на порт принимающей программы не только локального, но и другого компьютера. Таким образом, сервер SSL и сервер-получатель трафика могут находиться на разных компьютерах. Не плохо, чтобы сервер после расшифровки данных прятал IP-адрес клиента, с которого были отправлены данные, и это возможно, если указать опцию `-t`.

Во время установки пакета OpenSSL на вашем диске создаются сертификаты и пары ключей, которые используются для шифрования. Все это расположено в каталоге `/usr/share/ssl/`.

С помощью опции `-n` можно непосредственно указать протокол, с которым будет происходить работа. В настоящий момент поддерживаются POP3 (Post Office

Количество используемых алгоритмов исчисляется десятками. Наиболее распространенным является DES (Data Encryption Standard, стандарт шифрования данных). Я тоже отдаю предпочтение именно ему. О поддерживаемых алгоритмах можно узнать на сайте разработчиков или по команде: `man openssl`.

Параметр `-in` задает входной файл, который необходимо шифровать, а после ключа `-out` указывается имя файла, куда сохраняется результат.

При дешифровании в команду необходимо добавить ключ `-d`. Соответственно, аргументом параметра `-in` задается зашифрованный файл, а параметра `-out` — имя файла для сохранения результата:

```
/usr/bin/openssl алгоритм -d -in файл2 -out файл1
```

В качестве примера зашифруем список паролей `/etc/passwd` в файл `/home/passwd` по алгоритму DES. Для этого выполним команду:

```
/usr/bin/openssl des -in /etc/passwd -out /home/passwd
```

В ответ на эту директиву программа попросит вас указать пароль и затем повторить его ввод, чтобы исключить возможные ошибки.

Выполните команду: `cat /home/passwd` и убедитесь в том, что содержимое этого файла нечитаемо.

Для расшифровки файла выполните команду:

```
/usr/bin/openssl des -d -in /home/passwd -out /etc/passwd
```

Таким нехитрым способом мы можем безопасно хранить копию файла с паролями. К теме резервного копирования мы вернемся в *главе 13*, которая будет полностью посвящена этому вопросу.

5.2.4. Туннель глазами хакера

Хакеры могут использовать туннели для своих целей. Например, несколько лет назад мой компьютер был подключен к Интернету по технологии ADSL (Asymmetric Digital Subscriber, асимметричная цифровая абонентская линия). Это сейчас легко найти тарифы с безлимитным трафиком, а в начале 2000-х с этим были проблемы. В абонентскую плату входило всего 400 Мбайт трафика. Ну что такое 400 Мбайт? Для меня это день работы в экономном режиме, потому что общаться приходилось много, а из почтового ящика я иногда выкачиваю до 20 Мбайт в день. Превышение же лимита обходилось весьма дорого.

Как обойти ограничение и получить бесплатный трафик? Подобно большинству провайдеров, мой также предоставлял абсолютно бесплатный трафик со своих серверов, к которым можно было обращаться сколько угодно. Жаль, что на этих серверах ничего полезного для меня не было, но полезным оказался сам такой подход.

В мою абонентскую плату входило 10 Мбайт серверного пространства для создания визитной карточки в Интернете. Максимум, что получалось там разместить, — домашнюю страничку. Но мне было нужно не это. Главное, что трафик с этого сай-

та ничем не ограничивался. И если установить на сервере программу туннелирования, то можно организовать соединение, как показано на рис. 5.5.

Итак, хакер подключается через программу `stunnel` к бесплатному веб-серверу, который находится на площадке провайдера. Оттуда весь трафик перенаправляется на какой-нибудь прокси-сервер в Интернете или напрямую передается веб-сайтам. За это также платить деньги не надо, потому что связь с веб-сервером в обе стороны бесплатна.

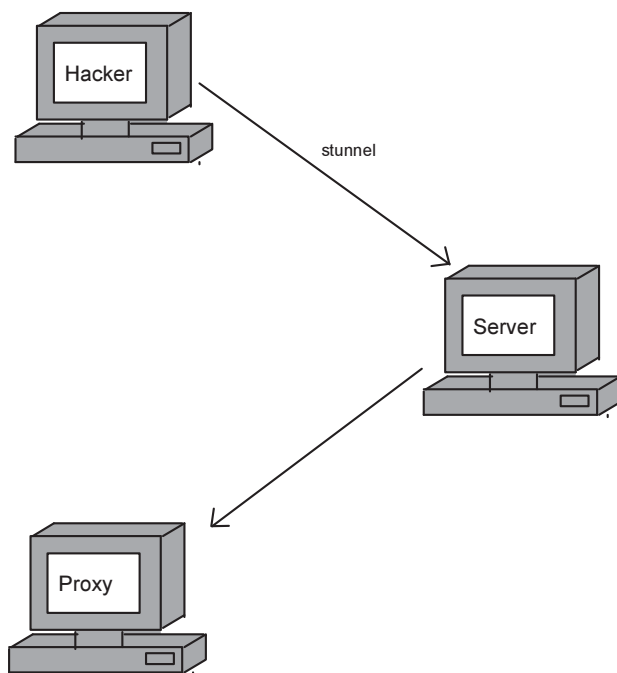


Рис. 5.5. Подключение к Интернету через веб-сервер

Таким образом, можно было скачивать не 400 Мбайт трафика, а целые гигабайты, и все бесплатно. Не знаю, остались еще где-то лимитированные тарифы, но в Канаде (где я живу) они до сих пор еще есть.

Еще один способ нестандартного использования туннелирования — расширение возможностей сетевого доступа. Например, в некоторых локальных сетях может быть запрещен какой-либо протокол доступа в Интернет. Мне довелось работать в организации, где было разрешено обращение к веб-сайтам только по протоколу HTTP. Все остальное было запрещено. Руководство мотивировало это тем, что пользователи не должны иметь возможность передачи файлов в Интернет, чтобы нельзя было отправить куда-либо секретную информацию.

Как обойти такое ограничение? Снова ставим туннель на веб-сервере, и можно использовать любой другой протокол, пряча весь трафик в HTTP, откуда туннель уже направляет данные на нужные порты с нужным протоколом.

Например, нам нужен доступ к FTP. На каком-нибудь сервере в Интернете на порту 80 (доступ к которому разрешен, потому что это стандартный порт для веб-сервера)

ставим сервер туннеля и настраиваем его на подключение к нужному FTP-серверу. А на своем компьютере устанавливаем клиент. С помощью клиента соединяемся с портом 80 своего сервера и можем передавать данные, которые будут перенаправляться на нужный FTP-сервер.

Такие туннели уже не нуждаются в шифровании и могут быть реализованы с помощью несложных программ — например, сценариев на языке Perl. Для передачи пакетов не обязательно использовать HTTP. Подойдет практически любой протокол на основе TCP.

Как видите, абсолютно безопасная на первый взгляд и даже предназначенная для защиты технология превращается в средство взлома ограничений, которыми «наградил» вас администратор.

Если вы обладаете хорошими знаниями программирования на PHP или Perl, то можете написать веб-сценарии, которые на сервере будут обращаться к нужным вам ресурсам, чтобы работать с необходимым сервером через веб-браузер. Получится что-то похожее на работу с почтой через сеть. Эта возможность есть на большинстве почтовых сервисов, и вам она должна быть знакома на практике.

5.2.5. Шифрование домашнего каталога

Я давно уже не использовал Linux в качестве десктопной системы, а только в качестве серверной. Когда ОС установлена на сервере, то обычно этот сервер находится в яйце, яйцо в курице, курица в ларце... Ой, это не та книга, но смысл тот же самый.

Серверы — это обычно стационарные компьютеры, которые чаще всего сейчас размещены в специальных стойках, стойки собраны в специальных помещениях с надлежащим охлаждением, помещения находятся в специальных центрах данных, которые обычно под хорошей защитой. Я не знаю, имеет ли смысл шифровать диски на серверах и делает ли это кто-то? Я никогда не шифрую диски на серверах, потому что уверен в средствах защиты центров данных и уверен, что к стойкам с серверами доступ получают только те, кто реально имеет на это право.

А вот для ноутбуков с десктопной ОС шифрование очень даже необходимо. Как мы уже увидели ранее, восстановить пароль в ОС Linux не так уж и сложно при наличии физического доступа, если подключить диск к другому компьютеру. От подключения к другому компьютеру может спасти шифрование диска. Десктопные компьютеры иногда воруют, у меня на моей первой работе был случай в 1996 году, когда из офиса на 6-м этаже (самый верхний) украли новенький компьютер на процессоре Intel Pentium. Как же жалко было его.

Ну а в случае с ноутбуками шифрование дисков — это просто обязательная функция. Ноутбуки воруют, забывают в публичных местах, и данные на них спасти может только шифрование.

Хотя на протяжении всей книги я описываю серверную ОС и при этом считаю, что шифрование принесет больше пользы ноутбукам, мы все же рассмотрим здесь шифрование, потому что это все же относится к безопасности, которая является центральной темой этой книги.

Я рекомендую полное шифрование всего диска, но есть способ проще — зашифровать только пользовательские файлы из каталога `home`. Дело в том, что шифровать исполняемые файлы нет смысла, а очень часто самое важное хранится именно в пользовательских данных.

Для шифрования всего домашнего каталога пользователя можно использовать программу `ecryptfs`. Допустим, что у вас в системе есть пользователь `username`, тогда его домашний каталог можно зашифровать следующей командой:

```
sudo encryptfs-migrate-home -u username
```

Если программа `ecryptfs` у вас не установлена, то ее можно установить так:

```
sudo apt-get install encryptfs-utils cryptsetup
```

Отсутствие на компьютере программы `ecryptfs` — не единственная причина, почему команда ее запуска может не выполниться: файлы в домашнем каталоге пользователя в этот момент использоваться не должны. Лучше всего, если пользователь полностью выйдет из системы.

Если команда миграции выполнена успешно, то вы должны увидеть инструкцию, согласно которой пользователь должен войти в систему, и это нужно сделать до ее перезагрузки. Если он сможет это сделать и сможет работать с файлами, то все прошло успешно.

Чтобы сделать все наиболее безопасно, сам пользователь должен выполнить команду:

```
encryptfs-unwrap-passphrase
```

Команду эту также нужно выполнить сразу после миграции и не затягивать с ней. Она не требует прав `sudo`, что вполне логично, — ведь простые пользователи не могут и не должны иметь права повышать свои привелегии. Смысл команды — создать фразу для шифрования.

Шифрование пользовательских файлов делает их практически защищенными. Осталась одна возможность, которая может стать причиной утечки зашифрованной информации, — файл подкачки. Если какая-либо пользовательская информация попала в память и ОС сбросила эту страницу памяти на диск в файл подкачки, а хакер смог прочитать этот файл, то это станет для вас серьезной проблемой.

Поэтому рекомендуется зашифровать и файл подкачки, для чего администратор системы должен выполнить команду:

```
encryptfs-setup-swap
```

Прежде чем программа завершит работу, она запросит подтверждение, потому что эта операция нарушит сон вашего ноутбука, — перестанет работать функция гибернации, так как ОС не сможет восстановить работу из зашифрованного файла подкачки.

Это недостаток шифрования пользовательских папок по сравнению с полным шифрованием диска.

5.3. Протокол SSH

Мы уже говорили о том, что Telnet не подходит для удаленного управления сервером, потому что далеко не безопасен. А желание и потребность в таком управлении есть. В больших сетях, как правило, задействованы несколько серверов, и бегать от одного монитора к другому накладно и неудобно. Любой администратор хочет сидеть за одним компьютером и удаленно управлять всем комплексом, используя при этом безопасные каналы связи.

Во время сеансов удаленного управления администратор передает по сети много конфиденциальной информации (например, пароли root), которая ни в коем случае не должна быть видна прослушивающим утилитами. Для обеспечения безопасной связи создано множество различных программ, но самой популярной стала SSH, которая сейчас поставляется в большинстве дистрибутивов Linux.

На Windows-серверах я часто использую утилиту Cygwin, в которой у меня работают различные скрипты. Cygwin — это аналог командной строки Linux под Windows, в которой есть практически все Linux-команды. Не знаю, как сейчас, но раньше в Cygwin, кажется, не было команды `sudo`, и чтобы получить права администратора, достаточно было просто соединиться по SSH-протоколу с локальной машиной. Открываем Cygwin, соединяемся с локальной машиной от пользователя root и выполняем команды уже от его имени.

Преимущество протокола SSH заключается в том, что он позволяет удаленно выполнять команды, но при этом требует аутентификации и шифрует канал связи. Важным моментом является и то, что даже пароли при проверке подлинности пользователя передаются в зашифрованном виде.

В настоящее время существуют две версии протокола SSH — с номерами 1 и 2. Вторая версия использует более стойкий алгоритм шифрования и исправляет некоторые ошибки предыдущей версии. В Linux сейчас поддерживаются обе версии.

Любопытно, что в ОС Linux за протокол SSH отвечает OpenSSH, для которого родной платформой можно считать другую UNIX-систему — OpenBSD — и который клонировался во все UNIX-платформы, в том числе и в Linux. Но даже сейчас в конфигурационных файлах можно иногда увидеть в комментариях имя OpenBSD.

Протокол SSH требует для работы настройки сервера и клиента. Сервер ожидает подключения и выполняет директивы пользователя, а с помощью клиента вы осуществляете соединение с сервером и посылаете запросы на выполнение команд. Таким образом, для нормальной работы вы должны правильно сконфигурировать обе части протокола.

5.3.1. Конфигурационные файлы

Все настроечные файлы протокола SSH находятся в каталоге `/etc/ssh`. Если конфигурационного файла нет, значит, соответствующий пакет не установлен, и следует выполнить команду:

```
sudo apt-get install ssh
```

В каталоге `/etc/ssh` можно увидеть следующий набор файлов:

- ❑ файл конфигурации SSH-сервера — `sshd_config`;
- ❑ файл конфигурации SSH-клиента — `ssh_config`;
- ❑ файлы ключей для различных алгоритмов:
 - `ssh_host_dsa_key`;
 - `ssh_host_dsa_key.pub`;
 - `ssh_host_key`;
 - `ssh_host_key.pub`;
 - `ssh_host_rsa_key`;
 - `ssh_host_rsa_key.pub`.

Почему так много файлов с ключами? Просто SSH работает с разными алгоритмами шифрования и поддерживает два наиболее популярных и криптостойких алгоритма: DSA (`ssh_host_dsa_key`, `ssh_host_dsa_key.pub`) и RSA (`ssh_host_rsa_key` и `ssh_host_rsa_key.pub`).

Замечу, что первое сокращение — DSA — означает Digital Signature Algorithm (алгоритм цифровой подписи), тогда как второе — RSA, несмотря на схожесть написания, состоит из первых букв имен основателей этого алгоритма шифрования: Ronald Rivest, Adi Shamir и Leonard Adleman.

Оставшиеся два файла: `ssh_host_key` и `ssh_host_key.pub` — хранят ключи для первой версии SSH. Для каждого алгоритма требуется по два файла: с расширением `pub` — хранит публичный (открытый) ключ, без расширения — содержит приватный (закрытый) ключ.

С помощью открытого ключа можно зашифровать данные и отправить их на сервер, но для расшифровки нужен только закрытый ключ, который не может быть подобран простыми алгоритмами. Он должен иметься только у вас, его необходимо беречь и никому не показывать.

5.3.2. Основные параметры конфигурации сервера SSH

Давайте теперь рассмотрим содержимое файла конфигурации SSH-сервера (`sshd`). Файл достаточно большой, и для экономии места я привел только небольшую его часть (листинг 5.1).

Листинг 5.1. Файл конфигурации `sshd`

```
#Port 22
#Protocol 2,1
#ListenAddress 0.0.0.0
#ListenAddress ::
```

```
# HostKey for protocol version 1
# Ключ для первой версии протокола
#HostKey /etc/ssh/ssh_host_key
# HostKeys for protocol version 2
# Ключи для второй версии протокола
#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_dsa_key

# Lifetime and size of ephemeral version 1 server key
# Время жизни и размер регенерируемого серверного ключа версии 1
#KeyRegenerationInterval 3600
#ServerKeyBits 768

# Logging
# Ведение логов
#obsoletes QuietMode and FascistLogging
#SyslogFacility AUTH
SyslogFacility AUTHPRIV
#LogLevel INFO
```

Рассмотрим основные параметры, которые вам могут пригодиться:

- ❑ **Port** — порт, на котором ожидаются подключения. По умолчанию — 22. Некоторые администраторы любят изменять значение по умолчанию, перенося сервер на другой порт. В какой-то степени это оправданно. Например, если у вас нет веб-сервера, то можно поместить SSH на порт 80. Хакеры будут думать, что это веб-сервер, и не станут его ломать;
- ❑ **Protocol** — поддерживаемые протоколы. Обратите внимание, что первым идет число 2, а затем 1. Это значит, что сервер будет сначала пытаться подключиться по второй версии протокола, и только затем по первой. Я рекомендую убрать знак комментария с этой строки и удалить число 1, чтобы использовалась только последняя версия. Уже давно пора обновить клиентское программное обеспечение и перейти на более безопасные технологии. Зацикливание на старых программах приносит только убытки;
- ❑ **ListenAddress** — адрес для прослушивания подключения. У вашего сервера может быть несколько сетевых карт. По умолчанию идет прослушивание всех интерфейсов. Вы должны указать только те, с которых будете подключаться по SSH. Например, очень часто одна сетевая карта смотрит во внутреннюю сеть, а другая — в Интернет. Если вы будете подключаться по SSH-протоколу лишь из внутренней сети, то следует прослушивать только этот адрес (задается в формате `адрес:порт`). Разрешается включать несколько таких записей, чтобы указать требуемые интерфейсы;
- ❑ **HostKey** — путь к файлам, содержащим ключи шифрования. Нужно прописывать только закрытые ключи, которые использует сервер для расшифровки пакетов;
- ❑ **KeyRegenerationInterval** — интервал времени, через который сервер обновляет ключи. В версии 1 во время сессии ключи могут регенерироваться. Это позволя-

ет сделать невозможным раскрытие пакетов за счет смены ключей, которые по умолчанию меняются каждые 3600 секунд. Если установить 0, то регенерации не будет. Так как мы отказались от первой версии протокола (см. параметр `Protocol`), то этот атрибут не влияет на работу;

- ❑ `ServerKeyBits` — длина серверного ключа. По умолчанию установлено 768, минимальное значение — 512;
- ❑ `SyslogFacility` — тип сообщений, которые будут сохраняться в журнале;
- ❑ `LogLevel` — уровень событий, которые будут попадать в журнал. Возможные уровни соответствуют системным, которые мы будем рассматривать в *разд. 12.5*;
- ❑ `LoginGraceTime` — интервал времени, в течение которого пользователь должен ввести правильный пароль, иначе соединение будет разорвано;
- ❑ `PermitRootLogin` — флаг, определяющий, разрешено (`yes`) или запрещено (`no`) входить в систему по протоколу SSH под пользователем `root`. Мы уже говорили, что `root` — это бог в системе, и его возможности нужно использовать аккуратно. Если в систему нельзя входить с такими правами, то и по SSH тем более. Срочно меняйте этот параметр на `no`;
- ❑ `StrictModes` — флаг, регламентирующий необходимость проверки состояния файлов и их владельцев, пользовательских файлов и домашнего каталога до ввода пароля. Желательно установить `yes`, потому что многие начинающие пользователи делают все свои файлы доступными для записи;
- ❑ `RSAAuthentication` — разрешена ли аутентификация по алгоритму RSA. Параметр действует для первой версии протокола;
- ❑ `PubkeyAuthentication` — дозволена ли аутентификация по публичному ключу. Параметр действует для второй версии протокола;
- ❑ `AuthorizedKeysFile` — файл с публичным ключом, который может использоваться для аутентификации;
- ❑ `RhostsAuthentication` — флаг, разрешающий аутентификацию по файлам `$HOME/.rhosts` и `/etc/hosts.equiv`. По умолчанию стоит `no`, и без особой надобности менять этот параметр не стоит, потому что это небезопасно;
- ❑ `IgnoreRhosts` — флаг, запрещающий читать файлы `~/.rhosts` и `~/.shosts`. Без необходимости значение лучше не изменять, потому что это может повлиять на безопасность;
- ❑ `AuthorizedKeysFile` — файл для хранения списка авторизованных ключей. Если пользователь входит в систему с имеющимся в этом файле ключом, то его пустят автоматически без ввода дополнительных паролей;
- ❑ `RhostsRSAAuthentication` — флаг, регламентирующий использование при аутентификации ключа хоста из каталога `/etc/ssh/ssh_known_hosts`. Параметр применяется в первой версии SSH;
- ❑ `IgnoreUserKnownHosts` — параметр, указывающий на необходимость игнорирования пользовательских списков доверенных компьютеров. Если он равен `no`, то

необходимо доверять компьютерам из списка `~/.ssh/known_hosts` при аутентификации `RhostsRSAAuthentication`. Не верьте никому, поэтому параметр лучше всего изменить на `yes`;

- ☐ `PasswordAuthentication` — требование пароля. Если значение равно `yes`, то будет требоваться пароль. При использовании авторизации через ключи параметр можно отключить;
- ☐ `PermitEmptyPasswords` — разрешение пустых паролей. По умолчанию установлено `no`, что запрещает использование пустых паролей, и изменять это значение не стоит;
- ☐ `KerberosAuthentication` — использование проверки подлинности по протоколу Kerberos. В последнее время именно эта аутентификация набирает большую популярность благодаря своей безопасности;
- ☐ `KerberosOrLocalPasswd` — если пароль Kerberos не был принят, то включается проверка локального файла паролей из файла `/etc/shadow`;
- ☐ `KerberosTicketCleanup` — удаление билета Kerberos из кэша при выходе из системы;
- ☐ `Banner` — позволяет указать файл, в котором находится текст приветствия, отображаемого пользователям.

5.3.3. Параметры доступа к серверу *sshd*

Кроме директив, приведенных в листинге 5.1, можно использовать следующие:

- ☐ `AllowGroups` — позволить вход в систему только пользователям указанных групп (указываются через пробел в одной строке);
- ☐ `AllowUsers` — разрешить вход в систему пользователям, имена которых указаны через пробел;
- ☐ `DenyGroups` — запретить вход в систему пользователям указанных через пробел групп;
- ☐ `DenyUsers` — запретить вход в систему пользователям, имена которых указаны через пробел. Этот параметр бывает удобен, когда дано разрешение на вход группе, но нужно отказать в подключении к SSH-серверу одному из ее пользователей.

Я рекомендую вам явно прописать группы или имена пользователей, которые могут входить в систему по SSH.

5.3.4. Конфигурирование клиента SSH

Настройки SSH-клиента содержат еще меньше параметров. В файле `/etc/ssh/ssh_config` находятся глобальные настройки для всех пользователей в системе. Но вы можете для любого из них переопределить произвольный параметр в файле `.ssh_config` из каталога пользователя. В листинге 5.2 приведено содержимое глобального файла настроек без комментариев.

Листинг 5.2. Конфигурационный файл /etc/ssh/ssh_config

```
# Site-wide defaults for various options
# Значения некоторых параметров для всех пользователей по умолчанию

# Host *
#   ForwardAgent no
#   ForwardX11 no
#   RhostsAuthentication yes
#   RhostsRSAAuthentication yes
#   RSAAuthentication yes
#   PasswordAuthentication yes
#   FallBackToRsh no
#   UseRsh no
#   BatchMode no
#   CheckHostIP yes
#   StrictHostKeyChecking ask
#   IdentityFile ~/.ssh/identity
#   IdentityFile ~/.ssh/id_rsa
#   IdentityFile ~/.ssh/id_dsa
#   Port 22
#   Protocol 2,1
#   Cipher 3des
#   Ciphers aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour
#   EscapeChar ~
Host *
Protocol 2,1
```

Некоторые из этих параметров мы уже видели при рассмотрении серверного конфигурационного файла. Здесь также можно увидеть параметр `Protocol`, в котором указываются используемые версии SSH. В рассматриваемом случае не стоит запрещать версию 1. На безопасность клиента это не повлияет, зато не будет проблем при подключении к серверу, который работает только на такой версии. Я надеюсь, что это будет не ваш сервер 😊.

Вот характерные для клиента команды:

- ☐ `Host` — сервер, к которому относятся следующие настройки;
- ☐ `CheckHostIP` — разрешение проверки IP-адреса с указанными в файле `known_hosts` адресами;
- ☐ `Compression` — разрешение (`yes`) или запрет (`no`) использования сжатия данных;
- ☐ `KerberosAuthentication` — разрешение (`yes`) или запрет (`no`) использования аутентификации по протоколу Kerberos;
- ☐ `NumberOfPasswordPrompts` — количество попыток ввода пароля. Если пароль не введен верно, то соединение разрывается;

- ☐ IdentityFile — имя файла, содержащего закрытые ключи пользователя;
- ☐ PasswordAuthentication — режим аутентификации по паролю.

5.3.5. Пример работы клиента SSH

Теперь рассмотрим на примере, как можно подключиться к удаленному серверу. Для этого служит команда:

```
ssh пользователь@сервер
```

Например, чтобы подсоединиться к серверу flenovm под именем flenov, нужно выполнить следующую команду:

```
ssh flenov@flenovm
```

В ответ на это вы можете увидеть сообщение:

**The authenticity of host 'localhost(127.0.0.1)' can't be established
RSA1 key fingerprint is f2:a1:6b:d6:fc:d0:f2:a1:6b:d6:fc:d0.
Are you sure you want to continue connection (yes/no)?**

Этим сообщением программа информирует вас, что подлинность хоста не была установлена, и отображает снимок RSA-ключа. Для продолжения соединения необходимо набрать на клавиатуре *yes*. На экране появится уведомление:

Permanently added 'localhost' (RSA1) to the list of known hosts.

Здесь сообщается, что ключ добавлен в список известных хостов. Это значит, что в каталоге `.ssh/` вашего домашнего каталога появился (или обновился) файл `known_hosts` с ключом удаленной системы.

Затем предлагается ввести пароль пользователя. Если аутентификация прошла успешно, вы оказываетесь в системе и можете выполнять команды на удаленном компьютере, как будто бы сидите за его клавиатурой.

Подключаться к SSH-порту Linux вы можете и из Windows. Для этого существует множество различных клиентов.

5.3.6. Вход по ключу

Намного удобнее и безопаснее способ авторизации по ключу, а вход по паролю может быть даже и вовсе заблокирован. Обращение к системе по SSH с помощью пароля не вполне безопасно. Злоумышленник может подсмотреть пароль, когда вы будете вводить его в другой программе. Тогда зачем шифровать SSH-соединение, если секретное слово может быть выявлено при работе с другими программами?

Каждое подключение должно иметь свои пароли. Но помнить их все очень сложно, поэтому лучше для авторизации использовать ключи, которые и так защищены дальше некуда. Нужно сделать только небольшие изменения в конфигурации.

Для начала надо создать новый ключ. Для этого служит команда `ssh-keygen`. У нее есть такие параметры:

- ❑ `-t` — тип ключа. Здесь можно указывать `rsa` или `dsa` для второй версии SSH или `rsa1` — для первой. Для примера мы чуть далее создадим ключ `rsa`;
- ❑ `-f` — файл, в котором будет сохранен закрытый ключ. Открытый ключ получит такое же имя, но с расширением `pub`;
- ❑ `-b` — длина ключа, которая не должна быть меньше 512. По умолчанию установлено значение 1024, оставим его и не будем указывать этот параметр.

Итак, для генерации ключа выполним команду:

```
ssh-keygen -t rsa -f ~/.ssh/myrsakey
```

Обратите внимание, что я указал сохранение ключа в подкаталоге `.ssh` своего домашнего каталога (об этом свидетельствует знак `~`). Это каталог, в котором SSH будет искать все настройки. Если вы еще не подключались к серверу, то этот путь и ключ отсутствуют. Для исправления ситуации нужно перейти в свой домашний каталог и создать в нем каталог `.ssh`:

```
cd /home/filenov  
mkdir .ssh
```

Если при генерации ключа не указывать файл для его сохранения, то по умолчанию он будет создан в каталоге `~/.ssh/` с именем `id_rsa` для RSA-шифрования. Для DSA-шифрования файл будет располагаться там же, но его имя будет `id_dsa`. Я специально задал имя, чтобы показать, как с ним работать.

Если программа запустилась успешно, то на экране вы должны увидеть следующее приглашение:

Generating public/private rsa key pair.

Enter passphrase (empty for no passphrase):

Здесь говорится, что начат процесс генерации публичного и закрытого RSA-ключей. Вам предлагается ввести кодовую фразу или оставить ее пустой. Я рекомендую лучше указать достаточно длинную фразу (не менее 10 символов, а лучше — целое предложение). После нажатия клавиши `<Enter>` вам предложат подтвердить комбинацию, чтобы исключить ошибки при вводе.

Если все прошло успешно, то вы должны увидеть следующее сообщение:

Your identification has been saved in `~/.ssh/myrsakey`.

Your public key has been saved in `~/.ssh/myrsakey.pub`.

В его первой строке нас проинформировали о том, что закрытый ключ сохранен в файле `~/.ssh/myrsakey`, а во второй — что открытый ключ сохранен в файле `~/.ssh/myrsakey.pub`.

Получив ключи, вы должны отправить файл `~/.ssh/myrsakey.pub` на удаленный компьютер, чтобы SSH-сервер мог использовать его для аутентификации. Передачу можно смело осуществлять через открытые каналы связи, потому что публичный ключ ничего не стоит без фразы (пароля), которую вы ввели, и без секретного ключа. Даже если хакер сможет получить файл `myrsakey.pub`, пользы от этого ему не будет никакой.

Администратор сервера должен добавить содержимое публичного ключа в файл `.ssh/authorized_keys`. Для этого можно выполнить на сервере следующую команду:

```
cat myrsakey.pub >> .ssh/authorized_keys
```

Теперь можно подключаться к серверу, используя публичный ключ для подтверждения личности. Но перед этим убедитесь, что в конфигурационном файле сервера включены следующие директивы:

```
RSAAuthentication yes  
PubkeyAuthentication yes
```

Для подключения к серверу выполните команду:

```
ssh -i ~/.ssh/myrsakey
```

С помощью параметра `-i` мы указываем файл публичного ключа. Если этого не сделать, то будет использоваться `id_rsa` — файл по умолчанию, его имя задает директива `IdentityFile` в конфигурационном файле SSH-клиента.

Теперь сервер будет запрашивать у вас не пароль, а слово (фразу), которое вы указали при генерации публичного ключа:

```
Enter passphrase for key
```

Если в конфигурационном файле SSH-сервера изменить параметр `PasswordAuthentication` на `no`, то пароль проверяться не будет, а связь станет устанавливаться только на основании ключей. Для обеспечения безопасной связи этого достаточно.

5.3.7. Защищенная передача данных

В состав пакета SSH входят еще две полезные программы: SFTP-server (FTP-сервер с поддержкой шифрования передаваемых данных) и SFTP-клиент (FTP-клиент для подключения к SFTP-серверу). Давайте посмотрим на последнюю строку файла конфигурации SSH-сервера: `/etc/ssh/sshd_config`:

```
Subsystem          sftp              /usr/libexec/openssh/sftp-server
```

Директива `Subsystem` определяет дополнительные сервисы. В приведенной строке как раз и запускается SFTP-server из пакета OpenSSH.

Работа с клиентом SFTP практически не отличается от работы с SSH-клиентом. Выполните команду: `sftp localhost`, и перед вами появится приглашение авторизации, которую мы рассматривали в *разд. 5.3.5*. Введя правильный пароль, вы окажетесь в командной строке FTP-клиента и можете передавать или принимать файлы, используя команды протокола FTP. Этот протокол мы будем подробно рассматривать в *главе 10*, а сейчас вам достаточно знать, что большинство команд схожи с директивами Linux для управления файлами. Основные команды протокола FTP приведены в *приложении 1*.

Попытайтесь сейчас воспользоваться клиентом SFTP для подключения к своей системе. Авторизовавшись, можно попробовать выполнить команду `ls` или `cd`, чтобы

убедиться в работоспособности программы. Для выхода из SFTP наберите команду `exit`.

Если вам необходимо передать на сервер или скачать с него секретную информацию (например, документы или файл паролей), то используйте для этого безопасное соединение по SFTP. Простые FTP-клиенты передают файлы в открытом виде (без шифрования), поэтому любой хакер сможет прослушать трафик и узнать информацию, которая поможет взломать ваш сервер.

В командной строке SFTP в основном приходится работать со следующими командами:

- ❑ `pwd` — показать путь к текущему каталогу;
- ❑ `ls` — показать файлы в текущем каталоге;
- ❑ `mkdir NAME` — создать папку с именем `NAME`;
- ❑ `cd PATH` — перейти в папку. В качестве `PATH` можно указывать полный путь к папке на удаленном сервере или только имя каталога, который является дочерним к текущему;
- ❑ `put FILE` — загрузить `FILE` с локального компьютера на удаленный;
- ❑ `get FILE` — скачать `FILE` с сервера. По умолчанию он будет скачиваться в ту папку, которая была активной перед активацией соединения.

Вы должны учитывать, что далеко не все FTP-клиенты поддерживают шифрование SSH, поэтому убедитесь в поддержке этого протокола со стороны вашего программного обеспечения. Например, если вы работаете в Windows и хотите организовать безопасное соединение с Linux для обновления файлов, можете воспользоваться программой WinSCP (www.winscp.net). Она обладает простым и удобным графическим интерфейсом, а любители движения Open Source смогут даже найти исходные коды программы.

5.4. Демон `inetd/xinetd`

Для того чтобы сервер смог обрабатывать запросы клиентов, программа должна быть постоянно загружена и связана с определенным портом. В этом нет ничего сложного, но иногда бывает невыгодно постоянно держать программу в памяти, если она большая, а работает очень редко. Если перед нами рабочий веб-сервер, то, конечно же, Apache должен работать на нем постоянно. А если это какая-то тестовая машина, то держать все запущенным нет смысла.

В ряде случаев лучше, когда один сервис в системе будет следить за портами и запускать необходимый сервис при обращении к определенному каналу. В ОС Linux такая возможность есть, и ее реализует демон `inetd` или его более новая версия — `xinetd`.

Как определить, что запускать? Для этого служит файл `/etc/services`, в котором находится список сервисов и их портов в следующем формате:

имя порт/протокол псевдоним

Здесь:

- ☐ Имя — название сервиса, который необходимо запускать;
- ☐ Порт — номер канала, который должен прослушиваться;
- ☐ Протокол — сервис `inetd` умеет работать с протоколами TCP и UDP, порты которых не пересекаются (они совершенно различны), так что необходимо в явном виде указывать протокол;
- ☐ Псевдоним — вымышленное имя для сервиса, которое можно не указывать.

Например, в файле `/etc/services` вы найдете следующие строки:

```
tcpmux 1/tcp    # TCP port service multiplexer
tcpmux 1/udp    # TCP port service multiplexer
rje     5/tcp    # Remote Job Entry
rje     5/udp    # Remote Job Entry
echo    7/tcp
...
...
ftp     21/tcp
ftp     21/udp   fsp fspd
```

Я специально выбрал эти строки, чтобы вы увидели варианты описания различных сервисов.

Если вы используете старый дистрибутив ОС Linux, то в нем, скорее всего, еще работает `inetd`. Но мы уже говорили, что устаревший дистрибутив не может быть безопасным, и с ним что-либо делать бесполезно. Лучшая защита — обновление, и тогда основным сервисом будет `xinetd`, который становится, если еще не стал, стандартом для всех.

Я рекомендую переход на `xinetd`, потому что в нем появилось много дополнительных возможностей, которые повышают удобство администрирования и безопасность. Например, в этот сервис встроены проверка всех удачных и неудачных соединений, возможность контроля доступа и даже предоставление его строго в определенное время.

По умолчанию `xinetd` может быть не установлен. Это довольно легко проверить. Выполните в командной строке: `xinetd`. Если команда завершится ошибкой, то требуется установка:

```
sudo apt-get install xinetd
```

5.4.1. Конфигурирование `xinetd`

Основной конфигурационный файл для `xinetd` — это `/etc/xinetd.conf`. В нем описываются настройки по умолчанию для запускаемых сервисов, а также каталог, в котором будут находиться конфигурационные файлы, влияющие на работу конкретных сервисов. Рассмотрим приведенное в листинге 5.3 содержимое этого файла.

Листинг 5.3. Файл конфигурации /etc/xinetd.conf

```
#
# Simple configuration file for xinetd
# Пример конфигурационного файла для xinetd
#
# Some defaults, and include /etc/xinetd.d/
# Некоторые параметры по умолчанию
# и подключение каталога /etc/xinetd.d/
defaults
{
    instances            = 60
    log_type              = SYSLOG authpriv
    log_on_success        = HOST PID
    log_on_failure        = HOST
    cps                   = 25 30
}

includedir /etc/xinetd.d
```

После ключевого слова `defaults` в фигурных скобках описываются настройки по умолчанию для всех сервисов. Любое из этих значений можно изменить для каждого отдельного сервиса.

Последняя строка подключает каталог `/etc/xinet.d`. В этом каталоге для каждой службы есть собственный конфигурационный файл, где можно изменить параметры. Имена файлов соответствуют названиям сервисов, а содержимое похоже на `/etc/xinetd.conf`. В листинге 5.4 приведено содержимое конфигурационного файла `/etc/xinet.d/telnet` для сервиса Telnet.

Листинг 5.4. Конфигурационный файл для сервиса Telnet

```
# default: on
# По умолчанию включен
# description: The telnet server serves telnet sessions; it uses \
# unencrypted username/password pairs for authentication.
# Описание: telnet-сервис обслуживает сессии telnet. Он использует
# незашифрованные имя пользователя и пароль для аутентификации
service telnet
{
    disable    = no
    flags      = REUSE
    socket_type = stream
    wait       = no
    user       = root
    server     = /usr/sbin/in.telnetd
    log_on_failure += USERID
}
```

Рассмотрим основные параметры, которые можно изменять:

- ❑ `disable` — параметр, установка которого в `yes` запрещает исполнение сервиса. Как уже говорилось, сервис `Telnet` небезопасен, и поэтому имеет смысл его запретить;
- ❑ `flags` — атрибуты выполнения сервиса;
- ❑ `socket_type` — тип используемого сокета. Для протокола `TCP` здесь должно быть значение `stream`, а для протокола `UDP` — `dgram`;
- ❑ `protocol` — используемый для передачи данных протокол (`TCP` или `UDP`);
- ❑ `server` — полный путь к запускаемой программе;
- ❑ `user` — права доступа. В большинстве случаев можно увидеть имя пользователя `root`. Это нормально, потому что в ОС `Linux` для работы с номерами портов менее 1024 необходимы права администратора. В настоящее время большинство сервисов понижают свои права в соответствии с установками;
- ❑ `instances` — максимальное количество одновременно работающих экземпляров программы;
- ❑ `log_type` — файл или системный журнал для записи событий;
- ❑ `log_on_success` и `log_on_failure` — информация, которая будет сохраняться в журнале при удачном и неудачном входе в систему соответственно. Здесь можно указывать значения: `PID`, `HOST` или `USER`;
- ❑ `per_source` — максимальное количество соединений от одного пользователя. Их может быть несколько, потому что пользователи любят максимально нагружать каналы и повышать скорость работы с помощью создания нескольких одновременно работающих соединений;
- ❑ `server_args` — аргументы, с которыми будет запускаться сервер.

При рассмотрении параметра `user` я упомянул о необходимости прав администратора для доступа к портам с номером менее 1024. Это действительно так, но зачем это нужно? Дело в том, что, не имея прав `root`, пользователь не сможет запустить сервер, который работает с портом от 1 до 1024. Такая защита необходима, потому что в этом диапазоне функционируют очень важные сервисы и их нельзя запускать простому пользователю.

Представьте себе, что хакер с правами простого пользователя смог бы запустить `FTP`-сервер, причем, возможно, ему удалось бы подменить конфигурационный файл при запуске, чтобы расширить свои права. Сделай он это, и у него появится возможность загружать на сервер файлы и скачивать их к себе на компьютер, что нежелательно.

5.4.2. Безопасность

Мы уже знаем, что программа `xinetd` позволяет определять права и время доступа к сервисам. Для этого в конфигурационном файле можно использовать три директивы: `no_access`, `only_from` и `access_time`.

Директива `no_access` запрещает доступ с указанных компьютеров. Например, следующая строка в конфигурационном файле закрывает доступ с адреса 192.168.1.1:

```
no_access 192.168.1.1
```

Если необходимо запретить доступ целой сети, то достаточно указать только ее номер. Например, если нужно закрыть вход всем компьютерам с адресами 192.168.1.x, где *x* может быть любым числом, надо использовать следующую строку:

```
no_access 192.168.1.
```

Обратите внимание, что в этом случае указанный IP-адрес состоит из трех октетов, разделенных точками, а не четырех; правда, после третьего числа указывается точка.

Для полного запрета доступа необходимо указать в конфигурационном файле следующую строку:

```
no_access 0.0.0.0
```

Теперь посмотрим, как можно предоставлять доступ с помощью директивы `only_from`. Она удобна тем, что изначально запрещает все, кроме указанных в качестве параметра адресов. Получается, что мы действуем от запрета. Указав эту команду без параметра, мы вообще запретим доступ:

```
only_from =
```

Я рекомендую включить эту строку в основной конфигурационный файл `/etc/xinetd.conf`, а потом для каждого отдельного сервиса в его собственном конфигурационном файле прописать разрешения. Например, давайте откроем доступ с адресов 192.168.1.2 и 192.168.1.19. Для этого добавляем в конфигурационный файл следующую строку:

```
only_from = 192.168.1.2 192.168.1.19
```

Можно разрешать доступ целым сетям:

```
only_from = 192.168.1.
```

А что, если всей сети разрешен доступ, а компьютеру с номером 1 запрещен? В этом случае можно прописать следующие две строки:

```
no_access = 192.168.1.1  
only_from = 192.168.1.
```

Запрет имеет больший приоритет, и даже несмотря на то, что у сети есть разрешение, компьютер с номером 192.168.1.1 из этой сети подключиться не сможет.

Теперь рассмотрим, как можно задать время доступа. Если вы настраиваете сервер, который будет работать в офисе компании, то вполне логичным будет разрешить подключение к нему только в рабочее время. Например, следующая строка делает сервисы доступными только с 8:00 до 18:00:

```
access_time = 8:00-18:00
```

В этом случае я бы увеличил второе значение до 19:00, потому что сотрудники часто задерживаются на работе, и не хочется, чтобы они дергали меня каждый день из-за доступа.

Функции обеспечения безопасности, встроенные в сервис `xinetd`, удобны и обладают достаточно мощными возможностями, хотя и дублируют права доступа, которые можно настраивать из файлов `/etc/hosts.allow` и `/etc/hosts.deny`. Мне больше нравится заниматься настройкой безопасности с помощью конфигурационных файлов сервиса `xinetd`, потому что параметры доступа хранятся в тех файлах, на которые они влияют.

5.5. Анализ безопасности дистрибутива

Мы уже говорили, что установка дистрибутива с параметрами по умолчанию далеко не всегда безопасна, потому что в ней могут быть проблемы с настройками, заданными прямо «из коробки». Такие настройки также могут привести к ошибке в конфигурировании.

Почему бы не автоматизировать анализ конфигурации настроек? Наверное, самой популярной утилитой такого плана является `Lynis`. Эта утилита поддерживает огромное количество ОС и разных дистрибутивов, среди которых, конечно же, есть `Ubuntu` и `CentOS`. Программа поддерживается не только на `Linux`, но и на `macOS`.

Установка выполняется, как обычно для `Ubuntu`, одной командой:

```
sudo apt-get install lynis
```

В `CentOS` соответственно используем команду:

```
yum install linux
```

Чтобы проверить систему на базовые проблемы конфигурации, выполняем команду:

```
sudo lynis audit system
```

Эта команда запускает базовый аудит. Чтобы выполнить все возможные проверки, то в конце команды нужно добавить ключ `-c`, что означает `check all`.

Я забыл выполнить команду на только что установленной системе, где у меня уже были добавлены `Apache` и `Samba`, но их я установил с настройками по умолчанию. В этих настройках я ничего не менял, и поэтому интересно посмотреть, что найдет система аудита в свеженькой `Ubuntu 20.04`. Результат работы только что приведенной команды аудита на моей системе показан на рис. 5.6.

В самом верху на экране можно увидеть надпись **Hardening index**. Я не уверен в точном переводе этой надписи, но в Интернете нашел, что ее переводят как Индекс Безопасности. Это именно индекс, а не процент, потому что, несмотря на возможные проблемы, которые нашла система аудита, не факт, что они эксплуатируемые, т. е. могут реально использоваться для взлома системы.

Во время работы программы информации на экран выводится много, и сразу ее всю проанализировать невозможно, поэтому все сохраняется в два файла:

```

Hardening index : 56 [#####          ]
Tests performed : 225
Plugins enabled : 1

Components:
- Firewall          [V]
- Malware scanner   [X]

Lynis Modules:
- Compliance Status [?]
- Security Audit    [V]
- Vulnerability Scan [V]

Files:
- Test and debug information : /var/log/lynis.log
- Report data                : /var/log/lynis-report.dat

=====
Notice: Lynis update available
Current version : 262   Latest version : 275
=====

Lynis 2.6.2

Auditing, system hardening, and compliance for UNIX-based systems
(Linux, macOS, BSD, and others)

2007-2018, CISofy - https://cisofy.com/lynis/
Enterprise support available (compliance, plugins, interface and tools)

=====

[TIP]: Enhance Lynis audits by adding your settings to custom.prf (see /etc/lynis/default.prf for
all settings)

mf1enov@mf1ubuntu:/etc$

```

Рис. 5.6. Результат выполнения на моей системе команды: `sudo lynis audit system`

- `/var/log/lynis.log` — здесь находится журнал проверки;
- `/var/log/lynis-report.dat` — здесь можно увидеть результат в виде отчета.

Результат проверки системы утилитой Lynis — достаточно субъективная вещь, потому что она выдает очень много рекомендаций по улучшению безопасности, которые могут быть на самом деле полезными для боевого сервера, и они действительно заслуживают внимания даже для домашних пользователей, да и для любого компьютера, подключенного к Интернету. Например:

- Install fail2ban to automatically ban hosts that commit multiple authentication errors — установите fail2ban, чтобы автоматически блокировать компьютеры, с которых было несколько ошибок авторизации. Полезная функция, особенно для некоторых типов серверов;
- Install debsums for the verification of installed package files against MD5 checksums — установите debsums, чтобы проверять установленные пакеты на изменение суммы MD5.

Среди проблемных моментов есть и такие, как политика паролей. По умолчанию в системе не были установлены сроки устаревания паролей. По умолчанию в ней также был установлен сетевой экран, но не установлены для него правила по умолчанию. Все это можно увидеть в отчете Lynis.

Она выдала много предупреждений, указывающих на необходимость зашифровать домашние каталоги различных пользователей, в том числе и пользователя root, который отключен. Это проблема для серверов, которые могут хранить пользовательские файлы. Если же рассматривается веб-сервер, то у него в домашних каталогах ничего не должно быть, а значит, шифровать что-то нет смысла.

Описывать все результаты аудита смысла нет, система выдала очень много информации, и по каждой проблеме дано короткое, но вполне понятное описание. При знании базового технического английского с ними легко справиться.

Первая часть отчета системы — это короткие результаты проверки, а во второй ее части есть раздел Lynis 2.6.2 Results, в котором каждая найденная проблема описана в следующем формате:

Короткое описание

Ссылка на подробную информацию

Например:

Enable auditd to collect audit information [ACCT-9628]

<https://cisofy.com/controls/ACCT-9628/>

Очень часто уже по короткой информации можно понять, что исправить, но если этого недостаточно, то можно открыть сайт по ссылке во второй строке и прочитать уже более подробно все доводы и причины для исправления найденной проблемы.

К сожалению, информация на сайте <http://cisofy.com/> приведена только на английском языке.



ГЛАВА 6

В стиле Samba

Изначально для обмена файлами между компьютерами служил протокол FTP (о нем мы подробно поговорим в *главе 10*). Но он неудобен, т. к. основан на технологии «клиент-сервер». Чтобы вы могли получить с чьего-то компьютера файл, на нем должен быть запущен FTP-сервер, а вам с помощью специальной программы, называемой FTP-клиентом, надо подключиться к этому серверу и скачать нужный файл. Сложность заключается не только в необходимости установки и настройки различных программ, но и в контроле доступа.

Чтобы не утруждать себя настройками FTP-сервера, многие стали организовывать в своих локальных сетях специализированные обменники. Для этого выделялся FTP-сервер с открытым доступом, который быстро превращался в мусорную корзину.

В Windows появился более удобный способ публиковать свои файлы — «Сетевое окружение», с помощью которого удаленный пользователь может зайти на любой компьютер и использовать его открытые ресурсы. Это действительно хорошая возможность, и пользователи привыкли к ней, несмотря на то что работа с открытыми ресурсами в первой реализации системы была небезопасной.

Чтобы пользователи Windows могли видеть сервер Linux в своем сетевом окружении и работать с ним как с Windows-машиной, был разработан пакет Samba (часто можно встретить сокращение smb), который состоит из двух программ: сервер позволяет публиковать локальные папки для всеобщего просмотра, а с помощью клиента вы можете подключаться к другим компьютерам и работать с их открытыми ресурсами.

Таким образом, с помощью Samba можно сделать легкодоступный и удобный в использовании файловый сервер. Раньше у меня на работе для этих целей использовался сервер Windows 2000, который параллельно работал как сервер баз данных. Но файловый архив занимает слишком много места, отнимает ресурсы сети и сервера. К тому же использовать дорогостоящую ОС Windows как файловое хранилище просто глупо — это то же самое, что вывозить мусор из дома на расстояние в 20 метров на «мерседесе». Поэтому было принято решение перенести файловый архив на отдельный физический сервер с бесплатной ОС.

Одно из мощнейших преимуществ Samba — возможность удаленного управления через SSH или SWAT (Samba WEB-based Administrative Tool, набор администратора через веб для Samba).

6.1. Конфигурирование Samba

В последней версии Ubuntu «самбу» убрали из установки по умолчанию серверной версии ОС, поэтому ее нужно установить:

```
sudo apt-get install samba
```

У меня сначала эта команда выполнялась с ошибкой — возможно, потому, что я использовал бета-версию Ubuntu, которая почти кандидат на релиз, и странно, что база данных пакетов оказалась в свежем дистрибутиве некорректна. Проблема решилась обновлением:

```
sudo apt-get update
```

Основным конфигурационным файлом для Samba является `smb.conf`, который можно найти в каталоге `/etc/samba/` (в некоторых дистрибутивах это может быть каталог `/etc`). Кроме него, в этом каталоге находится файл `lmhosts`, с помощью которого происходит сопоставление IP-адресов и имен компьютеров, — аналогично файлам `/etc/hosts` в Linux и `[Диск:]\Windows\System32\drivers\etc\lmhosts.sam` в Windows.

Дополнительно в этом же каталоге можно создать следующие файлы (некоторые из них могут существовать):

- `smbusers` — список пользователей, которым разрешено подключаться к серверу Samba;
- `smbpasswd` — пароли пользователей из файла `smbusers`.

Как видите, у Samba — свои конфигурационные файлы для хранения списка пользователей. Если вы создаете их вручную, то убедитесь, что права на чтение и запись установлены правильно. Файл должен быть доступен только администратору, т. е. владельцем может быть лишь `root` и никто иной.

Конфигурационный файл `smb.conf` содержит не так много директив, поэтому для удобства восприятия я привел в листинге 6.1 небольшой пример, который поможет вам увидеть общую структуру такого файла. В дальнейшем нам предстоит рассматривать другие серверы Linux, где настроек намного больше.

Листинг 6.1. Фрагмент конфигурационного файла `smb.conf`

```
[global]
# Основные директивы
workgroup = MYGROUP
server string = Samba Server

; hosts allow = 192.168.1. 192.168.2. 127.
load printers = yes
```

```
    printing = lprng
;   guest account = pcguest

# Директивы журнала
    log file = /var/log/samba/%m.log
    max log size = 0
    syslog only = no

# Директивы безопасности
    security = user
;   password server = <NT-Server-Name>
;   password level = 8
;   username level = 8
    encrypt passwords = yes
    smb passwd file = /etc/samba/smbpasswd
;   usershare max shares = 100

    unix password sync = Yes
    passwd program = /usr/bin/passwd %u
    passwd chat = *New*password* %n\n *Retype*new*password* %n\n
                  *passwd:*all*authentication*tokens*updated*successfully*
    pam password change = yes
;   username map = /etc/samba/smbusers

;   include = /etc/samba/smb.conf.%m
    obey pam restrictions = yes

# Настройка сокета
    socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
;   interfaces = 192.168.12.2/24 192.168.13.2/24
;   bind interfaces only = yes

# Настройка просмотра
;   remote browse sync = 192.168.3.25 192.168.5.255
;   remote announce = 192.168.1.255 192.168.2.44
;   local master = no
;   os level = 33
;   domain master = yes
;   preferred master = yes

# Работа с сервером
;   domain logons = yes
;   logon script = %m.bat
;   logon script = %U.bat
;   logon path = \\%L\Profiles\%U
;   wins support = yes

# WINS-сервер
    wins support = no
    wins server = w.x.y.z
```

```
dns proxy = no
name resolve order = lmhosts host wins bcast

# Отображение файлов
; preserve case = no
; short preserve case = no
; default case = lower
; case sensitive = no
```

Реальный файл в вашей системе будет намного больше, потому что он содержит множество комментариев с описаниями и примерами конфигурирования открытых каталогов. Здесь я все это удалил, чтобы вам было проще ориентироваться, когда мы станем рассматривать назначение команд.

В большинстве конфигурационных файлов Linux и ее программах для описания директив используется формат:

ИмяДирективы Значение

Имя директивы в этом случае должно состоять из одного слова и не может содержать пробелы. После имени ставится пробел, за которым идет значение директивы.

В Samba-сервере применяется несколько иной формат, приближенный к файлам настроек Windows:

ИмяДирективы=Значение

Значение директивы ставится после знака равенства. Таким образом, имя директивы может состоять из нескольких слов, содержать пробелы и различные символы (кроме знака равенства). Строки, начинающиеся с точки с запятой (;) или с решетки (#), рассматриваются как комментарии.

6.1.1. Основные настройки

Конфигурационный файл разбит на секции. Самой первой идет секция [global], в которой описываются глобальные настройки сервера. В ней можно увидеть следующие директивы:

- `workgroup = имя` — имя группы, в которую входит сервер. Когда в Windows вы входите в сетевое окружение, то можете увидеть все доступные ресурсы, разбитые на категории. В каждой группе могут быть свои компьютеры или серверы;
- `netbios name = имя` — имя, которое пользователи будут видеть в сетевом окружении для этого сервера, оно не должно совпадать с именем рабочей группы;
- `server string = описание` — свободный текст, который можно будет увидеть в поле **Description** (Комментарий) свойств сервера или окне сетевого окружения в режиме **Details** (Таблица). В этом поле вы можете поместить комментарий, описывающий содержимое сервера, — например: *Файловый архив Сергея*;
- `hosts allow = адреса` — указание (через пробел) IP-адресов или сетей, которым разрешен доступ к Samba-серверу. Например, чтобы открыть доступ всем ком-

пьютерам сети 192.168.1.x и одному компьютеру с адресом 192.168.2.2 из другой сети, надо написать следующую строку:

```
hosts allow = 192.168.1. 192.168.2.2
```

- ☐ `printcap name = файл` — файл описания принтеров, подключенных к системе. По умолчанию это `/etc/printcap`;
- ☐ `load printers = yes` — установка режима (*yes*) автоматического включения принтеров в список открытых ресурсов. Если в этом нет надобности, то укажите *no*;
- ☐ `printing = система` — тип системы печати. Здесь можно использовать одно из следующих значений: *bsd*, *sysv*, *plp*, *lpng*, *aix*, *hpux* или *qnx*.

6.1.2. Безопасность

В этом разделе мы рассмотрим директивы, которые напрямую или косвенно влияют на безопасность:

- ☐ `guest account = имя` — определяет учетную запись, с правами которой пользователь сможет входить в систему. Если ваш сервер не содержит секретной информации и используется для открытого обмена файлами, то можно завести гостевую учетную запись, иначе такой вход небезопасен;
- ☐ `log file = файл` — задает название файла-журнала. Например, можно написать: `/var/log/samba/%m.log`. Обратите внимание, что в имени присутствует символ процента, за которым следует буква *m*. Эта комбинация во время работы будет заменена именем пользователя, активность которого сохраняется. Так, например, для пользователя *robert* будет создан журнал `/var/log/samba/robert.log`;
- ☐ `max log size = n` — определяет максимальный размер журнала в килобайтах. Укажите значение 0, если ограничения не должно быть;
- ☐ `syslog only` — предписывает использовать для журналирования только *syslog*;
- ☐ `security = уровень` — определяет степень доступа. Параметр *уровень* может принимать одно из следующих значений:
 - *user* — доступ на уровне пользователя;
 - *share* — аутентификация на основе имени и пароля;
 - *server* — проверка пароля производится сторонним smb-сервером. Имя сервера, на котором хранятся пароли, задается с помощью директивы:


```
password server = ИмяСервера
```

 что позволяет держать пароли на другом smb-сервере, который и будет осуществлять проверку пароля;
 - *domain* — включает сервер в домен Windows NT, а пароль для доступа указывается в файле, определенном с помощью директивы:


```
smb passwd file = ИмяФайла
```

- `encrypt passwords = yes` — позволяет шифровать пароли, передаваемые по сети.

Эта опция требует пояснений, потому что могут возникнуть проблемы при авторизации с компьютеров с системой Windows. Суть в том, что Windows 95 вообще не шифровала пароли, и они передавались в открытом виде. Начиная с Windows 98, по умолчанию система стала шифровать пароли. Зашифрованный пароль передается по сети, и на сервере происходит его дешифровка. Если система ожидает зашифрованного пароля, а клиент посылает незашифрованный, то с соединением возникают серьезные проблемы.

Современные версии Windows шифруют пароль, но вы можете отключить эту возможность. Для этого в реестре нужно изменить параметр `EnablePlainTextPassword`, установив в нем значение 1. В Windows 9x этот параметр находится в реестре по адресу:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\VxD\ VNETSUP
```

Для Windows NT это:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Rdr\ Parameters
```

В Windows 2000 и более поздних версиях:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\
LanmanWorkstation\Parameters
```

Если соответствующего параметра не существует, его следует создать. Он должен иметь тип `Dword`.

Если возникли проблемы с входом в систему, то переключите систему в режим работы с незашифрованными паролями. В этом случае Samba использует для авторизации файлы `/etc/passwd` и `/etc/shadow`. Полученный открытый пароль шифруется по алгоритму MD5 и сравнивается со значениями из этого файла.

Если вы работаете в режиме с шифрованными паролями, то при авторизации будет использоваться файл `/etc/samba/smbpasswd` (это можно изменить с помощью директивы: `smb passwd file`). Такой файл необходим из-за различий в шифровании Windows и Linux;

СОВЕТ

Не применяйте открытые пароли без особой надобности. Не забывайте о существовании программ-снифферов, которые анализируют сетевой трафик и позволяют найти пароли, передаваемые по сети. Если они не зашифрованы, то злоумышленник сможет проникнуть в вашу систему.

- `smb passwd file = файл` — указывает на расположение файла с паролями. По умолчанию он находится в том же каталоге, где расположены конфигурационные файлы Samba;
- `ssl CA certFile = файл` — задает файл сертификата, необходимый для работы протокола SSL, гарантирующего безопасность передачи данных;
- `unix password sync = yes` — разрешает пользователям Windows менять пароль Samba, одновременно обновляя системные пароли в Linux. Если в этом нет не-

необходимости, установите значение параметра `no`. Для работы этой директивы нужно указать программы, которые будут изменять пароли (параметр: `passwd program`) и сообщения, появляющиеся перед пользователем (параметр: `passwd chat`). Приведу пример использования:

```
unix password sync = Yes
passwd program = /usr/bin/passwd %u
passwd chat = *New*password* %n\n *Retype*new*password* %n\n
*passwd:*all*authentication*tokens*updated*successfully*
```

Помимо этого, необходимо использовать директивы: `encrypt passwords` и `smb passwd file`;

- ☐ `username map = файл` — указание на файл, где хранится список пользователей сервера Samba для клиентов Windows 9x (о нем подробнее мы поговорим в *разд. 6.3*);
- ☐ `usershare max shares` — максимальное количество пользовательских каталогов, которые могут быть открыты для общего доступа;
- ☐ `usershare allow guests = yes` — если этот параметр включен, то гости системы смогут получить доступ к открытым ресурсам. По умолчанию этот параметр отключен, чтобы доступ к открытым ресурсам получали только авторизованные пользователи.

6.1.3. Сеть

В этом разделе мы изучим директивы настройки сетевого протокола:

- ☐ `include = файл` — позволяет подключить дополнительный конфигурационный файл, например, с другого компьютера или содержащий настройки для конкретного подключающегося компьютера. В последнем случае имя файла обычно задается в формате: `путь.%m`. Здесь `путь` — это начало полного имени файла, а `%m` — NetBIOS-имя компьютера. Например: `/etc/samba/smb.conf.robert`;
- ☐ `socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192` — задает параметры протокола и размер входного и выходного буферов. Здесь:
 - `TCP_NODELAY` — позволяет быстрее (без задержек) передавать данные;
 - `SO_RCVBUF` — размер принимающего буфера;
 - `SO_SNDBUF` — размер передающего буфера;
- ☐ `interfaces = интерфейсы` — если у вас установлены две сетевые карты, которые направлены на разные сети, то с помощью этой директивы можно позволить работать с Samba пользователям из какой-то конкретной или из обеих сетей;
- ☐ `bind interfaces only` — позволяет привязаться только к именованным интерфейсам или сетям, указанным в параметре `interfaces`. Если ваша сеть не защищена сетевым экраном, то настоятельно рекомендуется использовать этот пара-

метр, чтобы доступ к серверу Samba имели только компьютеры из вашей сети, а не любой удаленный пользователь из Интернета.

6.1.4. Замена сервера Windows

Если посмотреть на параметры, описанные в этом разделе, можно увидеть, что Samba способна полностью заменить Windows-сервер, и рабочие станции на базе Windows не заметят неудобств:

- ☐ `local master = yes` — позволяет сделать Samba-сервер основным браузером сети;
- ☐ `domain master = yes` — дает возможность сделать Samba-сервер главным браузером домена. Не устанавливайте значение `yes`, если в вашей сети уже есть контроллер домена Windows NT;
- ☐ `logon script = файл` — если директива `domain logons` включена, то в этом параметре можно указать файл bat-сценария, который будет выполняться на компьютере клиента при входе в систему. Сценарий может, например, задаваться в виде `%m.bat` (заменяется именем компьютера) или `%u.bat` (подменяется на имя пользователя);
- ☐ `logon path = путь` — определяет место хранения пользовательских профилей. При использовании этой директивы необходимо убрать комментарии с секции `[Profiles]`, которую можно найти в файле конфигурации по умолчанию.

6.1.5. Поддержка WINS и DNS

WINS (Windows Internet Naming Service, служба имен Интернета для Windows) — это сервис для сопоставления имен компьютеров и IP-адресов. База данных WINS чем-то напоминает DNS (Domain Name Service), только в ней хранятся NetBIOS-имена компьютеров, а DNS обслуживает доменные имена.

Для настройки работы через WINS служат следующие директивы:

- ☐ `wins support = yes` — разрешить использование WINS-сервера;
- ☐ `wins server = w.x.y.z` — IP-адрес WINS-сервера;
- ☐ `DNS Proxy = yes` — если эта опция разрешена, то нераспознанные NetBIOS-имена можно попытаться определить через DNS-сервер;
- ☐ `name resolve order` — последовательность, в которой будет производиться поиск адреса хоста по его имени. В конфигурационном файле можно найти следующий пример:

```
name resolve order = lmhosts host wins bcast
```

Если использовать это значение, то при поиске адреса система сначала будет искать имя хоста в файле `lmhosts`, затем в `hosts`, потом будет задействован файл `wins` и последним — `bcast` (broadcast, или широковещательный запрос).

6.1.6. Отображение файлов

В Linux и Windows применяются разные правила именования файлов. Например, в Linux названия чувствительны к регистру, а в Windows — нет. Это значит, что файлы DATA.TXT и data.txt в Windows будут восприняты как один и тот же файл, чего не скажешь о Linux.

Для решения этой проблемы есть несколько команд:

- ☐ `case sensitive = no` — не принимать во внимание различие в регистре;
- ☐ `default case = lower` — отображать все файлы в нижнем регистре;
- ☐ `preserve case = no` и `short preserve case = no` — запретить сохранение имен файлов с учетом регистра.

Если у вас в сети работают пользователи Windows-систем, то рекомендуется оставить указанные значения. Для однородной Linux-сети можно разрешить сохранять регистр.

6.2. Описание объектов

После определения основных параметров Samba-сервера можно описывать объекты, к которым может получать доступ пользователь. Это делается в отдельных секциях, которые идут после секции `[Global]`, рассмотренной в *разд. 6.1*.

6.2.1. Пора домой

Конечно же, любой пользователь захочет работать со своим каталогом. Для этого он должен иметь учетную запись в Linux, с которой и будет связан его каталог. Обращение к такому каталогу будет выглядеть так: `\\сервер\имя`, где сервер — это имя сервера или его IP-адрес, а `имя` — это имя пользователя, домашний каталог которого необходимо увидеть.

Чтобы разрешить пользователям работать с собственными каталогами, нужно описать раздел `[homes]`. Рассмотрим пример такой секции:

```
[homes]
comment = Home Directories
browseable = no
writable = yes
valid users = %S ; %S - строка с перечислением пользователей
create mode = 0750
directory mode = 0775
```

В этой секции могут присутствовать следующие директивы:

- ☐ `comment` — текстовый комментарий, который не влияет на работу;
- ☐ `browseable = no` — режим отображения ресурса в списке просмотра. Если указать `yes`, то в сетевом окружении будут видны папки пользователей;

- ❑ `writable = yes` — предикат записи в домашний каталог. При значении `no` создание и изменение файлов станет невозможным;
- ❑ `create mode = 0750` — права доступа для создаваемых файлов. В этом случае владелец файла имеет полные права, пользователи группы могут читать и выполнять его, а остальные — бесправные. Иногда следует понизить значение параметра до 740, чтобы пользователи группы могли только читать;
- ❑ `directory mode = 0775` — права доступа для создаваемых каталогов. В этом случае у пользователей группы тоже слишком высокие права, и я бы понизил их до 755, чтобы запретить им создавать файлы в новом каталоге. Остальные могут только просматривать каталог, но и это может оказаться лишним, и для большей безопасности лучшим решением будет значение 750;
- ❑ `valid users = пользователи` — список пользователей, которым разрешен доступ к домашним каталогам. Значения разделяются пробелом. По умолчанию доступ имеют все, но в реальных условиях это необходимо только некоторым. Поэтому я рекомендую в явном виде прописать имена тех пользователей, которые нуждаются в работе со своим домашним каталогом.

6.2.2. Доменный вход

Если вы настроили сервер Linux так, чтобы пользователи Windows могли входить в систему через `smb` , используя его как сервер домена, то необходимо убрать комментарии с секции `[netlogon]`:

```
; [netlogon]
; comment = Network Logon Service
; path = /usr/local/samba/lib/netlogon
; guest ok = yes
; writable = no
```

В этой секции также присутствуют комментарий и директива `writable`. Пользователи не должны иметь право записи в этот каталог, потому что здесь хранятся сценарии, которые выполняются при входе в систему. Файлы из этого каталога должен изменять только администратор.

Помимо этого, есть еще две команды:

- ❑ `path = путь` — полный путь к каталогу `netlogon`;
- ❑ `guest ok = yes` — разрешение гостевого входа.

Также нужно убрать комментарии в секции `[Profiles]`, которая выглядит следующим образом:

```
;[Profiles]
; path = /usr/local/samba/profiles
; browseable = no
; guest ok = yes
```

В этом каталоге хранятся профили пользователей, и он не должен быть виден в сетевом окружении Windows, поэтому директива `browseable = no` запрещает отображение.

6.2.3. Распечатка

Чтобы пользователям стали доступны принтеры, подключенные к Linux-серверу, нужно настроить следующую секцию:

```
[printers]
comment = All Printers
path = /var/spool/samba
browseable = no
guest ok = no
writable = no
printable = yes
```

Обратите внимание, что по умолчанию секция открыта, и зарегистрированные пользователи уже имеют доступ к принтерам. Если вы хотите, чтобы и «гости» смогли использовать принтер, то добавьте строку `public = yes`. Однако я не рекомендую это делать, потому что такая добавка предоставит пользователям лишнюю возможность для подшучивания. Например, в моей сети был случай, когда сотрудник рассылал на все принтеры с общим доступом разные картинки. Вроде безобидно, а работу тормозит, и бесполезно расходуются картриджи.

6.2.4. Общий доступ

Чаще всего на сервере необходим каталог, через который любой пользователь сможет обмениваться файлами с другими участниками сети. Для настройки такого каталога служит секция `[tmp]`:

```
;[tmp]
; comment = Temporary file space
; path = /tmp
; read only = no
; public = yes
```

По умолчанию секция закрыта, и для разрешения доступа к ней необходимо убрать комментарии. Обратите внимание на путь к открытому каталогу. Это `/tmp` — каталог для хранения временных файлов пользователей. С помощью директив `read only` (значение `no`) и `public` (значение `yes`) мы указываем серверу Samba, что каталог открыт, и в него могут записывать и из него читать файлы все пользователи.

Несмотря на удобства, которые предоставляет этот каталог, я рекомендую использовать его только в закрытых сетях. Если у вас есть выход в Интернет, то с помощью сетевого экрана необходимо ограничить доступ к Samba. Так как в каталог разрешено записывать любым пользователям, злоумышленник может записать в него свои файлы, которые помогут ему получить на сервере права администратора.

6.2.5. Личные каталоги

Все рассмотренные ранее разделы (секции) конфигурационного файла имеют устойчивые имена и предназначены для решения определенных задач. Но в некоторых случаях вы можете изменить имя раздела, и на работе сервера это не скажется. Однако я не рекомендую это делать, потому что в одной версии Samba все будет работать хорошо, а при обновлении программ сервера ситуация может измениться, и потом трудно будет найти ошибку, из-за которой Samba станет работать неверно.

Тем не менее вы можете создавать собственные разделы, в которых будете описывать некоторые права. Например, вам понадобится организовать общий каталог, в котором файлы будут доступны всем пользователям, но записывать туда смогут только пользователи определенной группы. Допустим, что в этом каталоге должны храниться какие-то картинки. Для решения такой задачи можно создать раздел [shareimages] следующим образом:

```
[shareimages]
comment = Share Images
path = /home/samba/images
public = yes
writable = yes
write list = @staff
printable = no
```

Для этого раздела задаются следующие директивы:

- ☐ path = /home/samba/images — каталог, который мы хотим открыть;
- ☐ public = yes — признак доступности;
- ☐ writable = yes — запись в каталог разрешена;
- ☐ write list = @staff — определяет группу staff, которая может записывать файлы. Для всех остальных каталог открыт только для чтения.

В таких объявлениях каталогов можно использовать любые директивы, которые мы рассматривали в этой главе.

6.2.6. CD-ROM

Отдельным разделом в файле конфигурации идут настройки прав доступа к CD-ROM. Вот пример этих настроек:

```
:[cdrom]
; comment = Samba server's CD-ROM
; read only = yes
; path = /cdrom
; guest ok = yes
```

Обратите внимание, что в начале каждой строки стоит символ точки с запятой. Я не случайно оставил их на месте, чтобы указать, что по умолчанию они есть (по крайней мере, у меня) — т. е. все опции раздела закомментированы. Если у вас этих

символов нет, то срочно подумайте о том, чтобы прошерстить всю систему очень подробно, — мало ли что еще ваш производитель разрешил по умолчанию.

Итак, все отключено, а значит, компакт-диск не будет виден по сети. Если нужно открыть к нему доступ, настройте следующие параметры:

- ☐ `comment = Samba server's CD-ROM` — комментарий;
- ☐ `read only = yes` — файлы доступны только для чтения;
- ☐ `path = /cdrom` — путь к диску. Если у вас два и более дисководов, нужно указать, какой из них вы хотите «расшарить», если же только один, то все равно нужно указать путь, куда он смонтирован;
- ☐ `guest ok = yes` — разрешить ли доступ к диску для «гостей» (неавторизованных в системе пользователей).

6.3. Управление пользователями

Для начала разберемся с именами пользователей. По умолчанию для доступа к серверу Samba применяются имена из системного файла `/etc/passwd`. Но вы можете завести отдельные записи Samba-сервера, которые будут соответствовать реальным, однако их можно будет использовать только для подключения к Samba, а не к системе.

Имена Samba описываются в файле `/etc/samba/smbusers`, расположение и название которого могут быть изменены директивой `username map` файла `smb.conf`. Содержимое файла `/etc/samba/smbusers` может быть, например, таким:

```
# Unix_name = SMB_name1 SMB_name2
root = administrator admin
nobody = guest pcguest smbguest
```

У списка имен несколько предназначений. Во-первых, с его помощью вы можете отразить имена, привычные для пользователей DOS и Windows, на учетные записи Linux. Например, в Windows максимальные права принадлежат пользователю Administrator, а в Linux это root. И во второй строке приведенного примера устанавливается соответствие имен `administrator` и `admin` пользователю `root`.

Во-вторых, использование этого файла позволяет аккумулировать несколько имен на одной учетной записи. Например, у вас есть группа пользователей, которым должны быть назначены одинаковые права. Для этого заводим в Linux только одну учетную запись `nobody` (под ней будут работать пользователи), а в Samba прописываем для нее нескольких пользователей: `guest`, `pcguest` и `smbguest` — под этими именами они будут входить в систему.

Несмотря на то что мы отразили имена пользователей `administrator` и `admin`, и это разные учетные записи, для них будет использоваться один пароль — назначенный пользователю `root`.

Информация о пользователях, которым разрешен доступ, хранится в файле `/etc/samba/smbpasswd`. Его расположение и имя могут быть изменены с помощью директивы `smb passwd file` файла `smb.conf`.

Рассмотрим пример содержимого файла:

```
flenov:0:813D6593C11F1173ED98178CA975D79:[UX      ]:LCT-41FA818F
robert:500:813D6593C11F1173ED98178CA975D79:[UX      ]:LCT-41FA818F
```

Сразу видно, что файл чем-то похож на `/etc/passwd`. Он также разделен двоеточиями на несколько колонок. Наиболее интересны из них первые три: имя пользователя, его UID в системе Linux и пароль.

Но добавлять пользователей вручную не очень удобно, потому что требуется зашифровать и прописать пароль, что не так уж и просто. Чтобы облегчить задачу, в пакет Samba включена утилита `smbpasswd`, которая принимает следующие параметры:

- `-a` — добавить пользователя в систему. Учетная запись должна уже существовать в файле `/etc/passwd`. Например, давайте пропишем Роберта, с которым мы уже не раз работали:

```
smbpasswd -a robert
```

В ответ на это программа попросит вас дважды ввести пароль. Указанная вами комбинация никак не влияет на системный пароль и используется только для доступа к Samba. Таким образом, пароли могут отличаться друг от друга. Я даже рекомендую сделать их различными. ОС Windows умеет запоминать пароли и хранить в своей системе, и версии Windows 9x делают это небезопасным способом. Если злоумышленник сможет украсть пароль на Samba, то он проникнет и в систему;

- `-x` — удалить пользователя. Чтобы исключить Роберта из системы, выполните команду: `smbpasswd -x robert`;
- `-d` — деактивировать пользователя. Если необходимо временно отключить доступ для пользователя, не удаляя его из системы, выполните команду `smbpasswd -d robert`. Давайте посмотрим на строку, соответствующую Роберту, после выполнения этой команды:

```
robert:500:813D6593C11F1173ED98178CA975D79:[DUX    ]:LCT-41FA818F
```

Обратите внимание, что в четвертой колонке в квадратных скобках появилась буква `D`. Она как раз и указывает на то, что запись деактивирована. Так вы легко можете определить, какие записи активны, а какие нет;

- `-e` — активировать пользователя. С помощью этой команды можно подключить деактивированного ранее пользователя: `smbpasswd -e robert`.

Если запустить `smbpasswd` без параметров, то вы сможете поменять свой пароль, и это можно будет сделать с текущими правами. Выполнение всех остальных команд требует прав администратора, поэтому их придется выполнять с использованием `sudo`.

Напоминаю, что файл `/etc/samba/smbpasswd` используется, если пароли передаются по сети в зашифрованном виде. В этом случае, чтобы предоставить доступ к Samba всем пользователям системы, необходимо для каждого выполнить команду: `smbpasswd`. Есть сценарии, которые автоматизируют работу, но их использование не

очень эффективно, потому что они не задают пароля и чаще всего перетаскивают всех пользователей, даже тех, кто не должен иметь доступа в систему. К таким пользователям относятся системные учетные записи типа bin, adm, deamon и др.

6.4. Использование Samba

Сервис Samba в основном создавался для пользователей Windows, но и поклонники Linux тоже оценили все преимущества этой технологии, тем более что ОС Linux управляет совместным доступом к файлам по сети не хуже Windows, а в чем-то даже лучше. Для работы с Samba из ОС Linux служит команда: `smbclient`.

Чтобы подключиться к серверу, необходимо указать как минимум два ключа: `-L` (адрес сервера) и `-U` (имя пользователя). В ответ на это программа запросит ввести пароль. Если вы не используете шифрование, то необходимо ввести системный пароль, в противном случае воспользуйтесь тем, который вы указали при переносе пользователя в файл `/etc/samba/smbpasswd` (при запуске команды `smbpasswd`).

Итак, выполните следующую команду для тестирования сервера:

```
smbclient -L localhost -U root
```

После ввода пароля пользователя `root` вы должны увидеть все открытые ресурсы сервера. Результат выглядит примерно так:

```
Domain=[MYGROUP] OS=[Unix] Server=[Samba 2.2.3a]
  Sharename      Type            Comment
  -----
  IPC$           IPC            IPC Service (Samba Server)
  ADMIN$         Disk           IPC Service (Samba Server)

  Server         Comment
  -----
  FLENOVM        Samba Server

  Workgroup      Master
  -----
  MYGROUP        FLENOVM
```

Вы должны учитывать, что в этом списке находятся не все каталоги. Если, например, у домашних каталогов из раздела `[homes]` файла конфигурации директива `browseable` установлена в значение `no` (см. *разд. 6.2.1*), таких каталогов видно не будет. Это вполне логично, потому что злоумышленнику нельзя давать возможность лицезреть имена каталогов, особенно если они соответствуют именам пользователей или содержат конфиденциальные данные. Никогда не изменяйте этот параметр, чтобы хакер не знал, что он должен сломать.

Для подключения к открытому ресурсу сервера нужно подать команду: `smbclient`, передав ей имя ресурса, которое задается в формате UNC (Universal Naming Convention, универсальное именование объектов) с применением следующего синтаксиса:

```
\\ИмяСервера\ресурс
```

Например, вы хотите подключить домашний каталог пользователя flenov. Его адресом будет:

```
\\192.168.1.1\\flenov
```

Но здесь надо сделать одно замечание: в Linux обратная косая черта (\) является служебным символом, поэтому каждый такой знак должен заменяться двумя символами \\ (т. е. каждый знак \ экранируется еще одним знаком \). Поскольку в начале UNC-имени идут два знака \\, то они заменяются на четыре обратных косых черты, и приведенный здесь адрес должен быть введен так:

```
\\\\192.168.1.1\\flenov.
```

Итак, для подключения к ресурсу выполняем команду:

```
smbclient \\\\192.168.1.1\\flenov
```

Если вы обращаетесь к ресурсу, который требует авторизации, необходимо указать имя пользователя, обладающего правами:

```
smbclient \\\\192.168.1.1\\flenov -U flenov
```

При удачном подключении к открытому ресурсу вы увидите такое приглашение:

Smb: \>

Теперь можно выполнять различные операции над файлами. Чтобы узнать, какие команды доступны, введите директиву help или знак вопроса (?). Команды, которые вы увидите, очень похожи на команды FTP (более подробно с FTP мы познакомимся в *главе 10*, см. также *приложение 1*). Для отключения от ресурса необходимо выполнить команду exit.

Большинство дистрибутивов Linux включают стандартный пакет Samba и ничего больше. А ведь в Интернете можно найти сторонние разработки, которые позволяют монтировать открытые в Samba ресурсы в файловую систему Linux так же, как дискету (флешку) или CD-ROM, или работать с общими ресурсами в графическом режиме, как это делается в Windows. Однако имейте в виду, что это не всегда безопасно.

6.5. Развитие Samba

Хотя сервис Samba имеет давнюю историю и вполне устойчивый код, он продолжает развиваться. В настоящий момент последняя стабильная версия программы имеет номер 3.5. Все версии, начиная с 3.0, способны подключаться к домену Active Directory, т. е. к службе каталогов, продвигаемой и развиваемой фирмой Microsoft со времени выпуска Windows 2000 Server. Однако в качестве контроллера домена Active Directory сервер Samba пока выступать не может. Сейчас разрабатывается версия 4.0 сервера Samba, которая будет лишена этого недостатка, и тогда можно будет полностью доверить Linux серверную часть сети, клиентские компьютеры которой работают под Windows.

Текущее состояние разработки Samba можно узнать на сайте проекта: www.samba.org.



ГЛАВА 7

Веб-сервер

Несмотря на то что изначально сеть создавалась для обмена файлами, с появлением первого браузера популярность WWW-страниц начала расти не по дням, а по часам. Сейчас уже трудно себе представить не только жизнь без Интернета, но и Интернет без веб-страниц.

Для того чтобы пользователь смог просматривать с вашего узла хотя бы простейшие веб-странички, необходим веб- или HTTP-сервер. Уже долгое время самым распространенным из них является Apache. Трудно точно сказать, какая доля серверов работает на Apache, но можно с уверенностью утверждать, что их больше половины. Хотя для Linux существуют и другие веб-серверы (например, TUX), когда говорят о веб-сервере под Linux, подразумевают именно Apache. Да и в Windows можно очень часто встретить именно Apache, хотя в последнее время большую популярность стал получать Microsoft IIS.

Apache — абсолютно бесплатный сервер и доступен не только для UNIX-систем, но и для ОС Windows. Я для разработки использую macOS и Apache, и когда я публикую сайт на серверах CentOS, мне абсолютно ничего менять не приходится. Официальный сайт разработчиков — **www.apache.org**, где, помимо HTTP-сервера, можно найти еще ряд других проектов. Почему этот сервер стал так популярен? Неужели так повлиял фактор нулевой цены? Бесплатность, конечно, соблазнительна, но качество превыше всего. Это не единственный бесплатный сервер, но Apache располагает громадным количеством возможностей и при этом обладает следующими немаловажными особенностями:

- ☐ безопасность — многие профессионалы относят его к разряду самых защищенных;
- ☐ надежность — в сочетании с ОС Linux этот веб-сервер может работать без перезагрузок годами, а если и случаются перезагрузки, то чаще всего для обновления системы;
- ☐ неприхотливость — работает с минимальной нагрузкой на систему (не требует особых ресурсов);

□ производительность — высокая скорость отклика и обработки запросов пользователей. Возможно, не самая высокая, но достаточная, чтобы справляться с большими потоками трафика.

На основе Apache строят серверы, которые доступны 99,9% времени в год. Многие корпорацииверяют ему свои важные данные. Да что там говорить, мои веб-сайты работают у хостера на серверах с Apache.

Единственный недостаток, который отмечают пользователи, — это сложность конфигурирования. Приходится редактировать текстовый файл, а так как настроечных параметров огромное количество, то это может стать утомительным занятием. При этом очень сложно гарантировать, что все настройки будут сделаны оптимальными и безопасными, — ведь очень легко упустить что-либо или банально опечататься. Да, в некоторых случаях конфигурационные файлы не так наглядны, как графические утилиты, но это дело привычки. Очень многие администраторы, наоборот, предпочитают именно текстовый формат настроек.

В этой книге мы не сможем рассмотреть все возможные параметры настроек Apache, потому что их слишком много. Мы коснемся только основных и сосредоточимся на том, что может сказаться на производительности и безопасности сервера. Рассматриваемого материала хватит в большинстве случаев, и лично мне никогда не приходилось изменять ничего из того, что мы опустим.

7.1. Основные настройки

Основные настройки сервера Apache располагаются в файле `/etc/httpd/conf/httpd.conf` (для некоторых дистрибутивов — в `/etc/httpd.conf`). С выходом Apache2 конфигурация переместилась в `/etc/apache2/`. Если раньше все настройки находились в одном файле `httpd.conf`, то сейчас они разделены на несколько файлов, что достаточно удобно, просто немного непривычно. В принципе можно так же обойтись и одним файлом, как и раньше.

Если вы не видите конфигурационных файлов, то, возможно, Apache у вас просто не установлен. Чтобы установить его в Ubuntu, выполняем:

```
sudo apt-get install apache2
```

А в CentOS:

```
yum install httpd
```

Теперь попробуйте проверить, доступны ли конфигурационные файлы:

```
ls /etc/apache2
```

Они должны быть доступны.

Начнем рассмотрение с основного конфигурационного файла: `/etc/apache2/apache2.conf`:

- `ServerRoot` — корневой каталог, в котором находятся файлы конфигурации и журналы;
- `Timeout` — предельное время ожидания в секундах для получения и отправки пакетов;

- ❑ `HostnameLookups` — флаг, отвечающий за проведение преобразования IP-адресов в доменные имена для логов и CGI-программ. Если установлено значение `on`, то IP-адрес клиента, запросившего данные с сервера, будет преобразован в доменное имя, иначе будет использоваться IP-адрес. Если в этом нет необходимости, для повышения быстродействия можно установить значение `off`;
- ❑ `User` и `Group` — имя пользователя и группа, с правами которых будет работать сервис. По умолчанию в Linux используется имя и группа `www-data`. Этот пользователь и группа должны обладать в системе минимальными правами, которые только необходимы для работы веб-сервера и его модулей. Ничего лишнего решать им нельзя.

В файле `apache.conf` вы не увидите имен, здесь будут только переменные вида:

```
User ${APACHE_RUN_USER}
Group ${APACHE_RUN_GROUP}
```

После знака доллара в фигурных скобках идут имена переменных. Реальные же значения этих переменных находятся в файле `/etc/apache2/envvars` (Environment Variables, переменные окружения). В этом файле переменные объявлены следующим образом:

```
export APACHE_RUN_USER=www-data
export APACHE_RUN_GROUP=www-data
```

Не думаю, что вам придется когда-либо менять эти параметры, так что эта информация просто к сведению. Ни один веб-сервер не должен работать с повышенными правами. Пользователь и группа `www-data` не обладают в системе ничем сверхъестественным и являются простыми пользователями, каковыми и должны оставаться;

- ❑ `ErrorLog` и `CustomLog` — местоположение журналов. Опять же, здесь вы увидите переменную `${APACHE_LOG_DIR}`, которую также можно задать в файле переменных окружения;
- ❑ `LogLevel` — степень подробности составления журнала. Можно указать одно из следующих значений: `emerg`, `alert`, `crit`, `error`, `warn`, `notice`, `info`, `debug`;
- ❑ `KeepAlive` — разрешение обрабатывать несколько запросов за одно соединение. По умолчанию этот параметр выключен (`off`), и для получения каждого файла требуется отдельное соединение. Это неэффективно и приводит к лишнему расходу ресурсов. Допустим, что пользователь запросил страницу с 10 картинками. В ответ на это браузер клиента откроет 11 соединений с веб-сервером (одно для получения документа HTML и 10 — для картинок документа). Если включить этот параметр, то за одно подключение может быть обработано несколько запросов.

Однако при включении параметра `on` сервер будет открывать соединения на долгий срок. Если злоумышленник откроет большое количество соединений с сервером и не закроет их, то тем самым сможет исчерпать все его ресурсы, и новые пользователи подключаться не смогут. Есть специальные программы, которые

позволяют открывать соединения и с небольшой скоростью скачивать данные¹. Медленная активность заставляет сервер не закрывать соединение, и таким образом с использованием лишь одного компьютера хакер может сделать недоступным даже мощный сервер. Так что параметр `KeepAlive` хоть и не является причиной атаки, но он ее упрощает.

О том, как защититься от этого вида атак, можно прочитать в отличной заметке: <https://community.qualys.com/blogs/securitylabs/2011/11/02/how-to-protect-against-slow-http-attacks> от Sergey Shekyan;

- ❑ `MaxKeepAliveRequests` — максимальное число запросов, которое может быть выполнено в течение одного соединения;
- ❑ `KeepAliveTimeout` — время ожидания очередного запроса от одного и того же клиента (в секундах). Если за указанный период запрос не поступит, то соединение завершается.

Рассмотрим теперь конфигурационный файл `/etc/apache2/conf-enabled/security.conf`:

- ❑ `ServerTokens` — характер информации об установленном программном обеспечении, возвращаемый сервером. По умолчанию при любом обращении к серверу он возвращает заголовок с подробной информацией о системе, которая включает версии Apache, ОС Linux и всех имеющихся модулей. Если хакер узнает, что на сервере установлена старая версия интерпретатора PHP (или любой другой программы), то на взлом уйдет намного меньше времени. Болтливые параметры необходимо отключать, а информацию о сервере прятать. `ServerTokens` может принимать одно из следующих значений:
 - `Full` — отображать полную информацию о сервере и установленных модулях, включая их версии. Самое опасное для сервиса — это использование именно этого параметра;
 - `Min` — показать минимум сведений (только название сервера и установленные модули). Иногда даже такой простой список без версий может сказать хакеру слишком много;
 - `ProductOnly` — только название сервера, в нашем случае Apache вернет свое имя (`apache`) без указания версий. Вот это как раз то, что нам нужно;
- ❑ `ServerSignature` — если этот параметр установлен в `on`, то сервер будет добавлять к сгенерированным страницам текстовую информацию, которая будет содержать имя сервера и виртуального хоста. В рабочих системах лучше отключать это, установив параметр в `off`.

7.2. Модули

При настройке сервиса Apache очень важным звеном являются модули. Раньше их нужно было описывать в конфигурационном файле `/etc/httpd/conf/httpd.conf` следующим образом:

¹ Для Linux можно скачать программы `slowhttptest` с сайта: code.google.com/p/slowhttptest/.

```
<IfDefine HAVE_PERL>
LoadModule perl_module modules/libperl.so
</IfDefine>
```

Сейчас все стало намного проще. Уже установленные и активированные модули можно увидеть в каталоге `/etc/apache2/mods-enabled`. Достаточно просто посмотреть, какие файлы там имеются:

```
ls /etc/apache2/mods-enabled
```

По умолчанию загружаются все установленные модули или те, которые вы видите в этом каталоге. У меня по умолчанию грузится PHP5, который стал очень популярным в связке LAMP. Я бы посоветовал загружать только то, что вам действительно нужно. Например, воспользовавшись ошибкой в сценарии PHP, злоумышленник может выполнять на сервере какие-либо команды. Хорошие сайты используют какой-то один язык веб-программирования: Perl, PHP или Python, и вы должны оставить только тот модуль, который необходим, а остальные неиспользуемые модули отключить — это значительно сузит возможности злоумышленника. Ничего не напоминает вам такая рекомендация? Да, это снова самое важное правило: лишняя программа — враг администратора и дополнительная «дверь» для хакера.

Я обратил внимание, что модуль `rewrite` не загружается. Давайте исправим это и заодно посмотрим, как можно включить новые модули. Для этого выполняем команду:

```
sudo a2enmod rewrite
```

Здесь `a2enmod` — это программа, которая помогает включать новые модули, а `rewrite` — это имя модуля. Проверьте еще раз каталог `/etc/apache2/mods-enabled` и убедитесь, что вы видите там файл `rewrite.load`.

Все установленные в системе модули (и активированные, и неактивированные) можно увидеть в каталоге `/etc/apache2/mods-available`.

Мы включили модуль, но изменения не вступят в силу, пока мы не перезагрузим Apache. Для этого выполняем команду:

```
sudo /etc/init.d/apache2 restart
```

7.3. Права доступа

Снова возвращаемся в основной файл конфигурации `/etc/apache2/apache2.conf`, но теперь поговорим о правах доступа к каталогам. Они описываются следующим образом:

```
<Directory /var/www/html>
    Order allow,deny
    Allow from all
</Directory>
```

или, например, так:

```
<Location /server-status>
  SetHandler server-status
  Order deny,allow
  Deny from all
  Allow from .your-domain.com
</Location>
```

Первое объявление разрешает доступ к определенному каталогу на диске (в нашем случае к `/var/www/html`), а второе — ограничивает доступ к виртуальному каталогу (в приведенном примере: **`http://servername/server-status`**), разрешая его только клиентам из домена **`your-domain.com`**.

Если вы знакомы с HTML (HyperText Markup Language, язык разметки гипертекста) или XML (Extensible Markup Language, расширенный язык разметки), то такое объявление будет вам знакомо и более или менее понятно без дополнительных пояснений. Для тех, кто не в курсе, я сделаю несколько пояснений на примере каталогов.

Итак, объявление начинается со следующей строки:

```
<Directory Путь>
```

В угловых скобках указывается ключевое слово `Directory` и путь к каталогу, права доступа к которому нужно изменить. Это начало описания, после которого идут команды, определяющие права. Блок заканчивается строкой:

```
</Directory>
```

Параметры доступа к каталогу могут быть описаны не только в основном конфигурационном файле, но и в файле `.htaccess`, который может находиться непосредственно в самом каталоге.

ПРИМЕЧАНИЕ

Что я имею в виду под основным файлом? В зависимости от дистрибутива, это может быть `/etc/apache2/apache2.conf` или `/usr/local/apache2/conf/httpd.conf` (какие конфигурационные файлы используются в каких дистрибутивах, можно увидеть на странице: **<https://wiki.apache.org/httpd/DistrosDefaultLayout>**). Далее я в основном буду ссылаться на `httpd.conf`, который является конфигурационным файлом по умолчанию (в том числе у меня на macOS), хотя в Ubuntu это файл `apache2.conf`.

Сам файл `.htaccess` требует отдельного рассмотрения (см. *разд. 7.5.1*), а сейчас вы должны только знать, что разрешения, указанные в конфигурации веб-сервера, могут быть переопределены.

При задании прав доступа следует иметь в виду, что Apache просматривает разрешения и запреты для всех каталогов, находящихся в иерархии выше запрошенного. При определении прав доступа к каталогу сперва определяются права доступа к корневому каталогу, которые затем могут быть переопределены правами доступа к любому подкаталогу, лежащему в пути к запрошенному документу. Это позволяет эффективно управлять правами.

Поскольку мы думаем о безопасности, надо вспомнить наше основное правило: запрещено все, что не разрешено явно. Поэтому имеет смысл поместить в основной конфигурационный файл (`httpd.conf` или `apache2.conf`) следующие строки:

```
<Directory />
    Order deny,allow
    Deny from all
</Directory>
```

которые запрещают доступ к любым каталогам, а затем открыть доступ только к тем каталогам, к которым он необходим. Теперь если на сервере создать новый каталог, а права доступа к нему не прописать, то доступ к нему будет закрыт, кроме тех случаев, когда каталог создан внутри каталога с уже открытым доступом.

А сейчас рассмотрим, как задаются права доступа. Для этого служат следующие директивы:

□ `Allow from` параметр — определяет, с каких хостов можно получить доступ к указанному каталогу. В качестве параметр можно использовать одно из следующих значений:

- `all` — разрешает доступ к каталогу всем;
- доменное имя — определяет домен, с которого можно получить доступ к каталогу. Например, можно указать `domain.com`. В этом случае только пользователи этого домена смогут получить доступ к каталогу через сеть. Если какая-либо папка содержит опасные файлы, с которыми должны работать только вы, то лучше ограничить доступ своим доменом или только локальной машиной, указав `Allow from localhost`;
- IP адрес — сужает доступ к каталогу до определенного IP-адреса. Это очень удобно, когда у вашего компьютера есть выделенный адрес и вы хотите обеспечить доступ к каталогу, содержащему администраторские сценарии, только для себя. Адрес может быть как полным, так и неполным, что позволяет ограничить доступ к каталогу определенной сетью;
- `env=ИмяПеременной` — разрешает доступ, если определена указанная переменная окружения. Полный формат директивы:

```
Allow from env= ИмяПеременной
```

□ `Deny from` параметр — запрещение доступа к указанному каталогу. Параметры такие же, как у команды `Allow from`, только в данном случае закрывается доступ для указанных адресов, доменов и т. д.;

□ `Order` параметр — очередность, в которой применяются директивы `allow` и `deny`. Могут быть два варианта:

- `Order deny,allow` — изначально все разрешено, потом проверяются запреты, но они могут быть обойдены при помощи разрешений. Рекомендуется это использовать только на общих каталогах, в которые пользователи могут самостоятельно закладывать файлы, — например, свои изображения;

- `Order allow,deny` — сначала все запрещено, вслед за этим проверяются разрешения, но и они могут быть обойдены запретами. Рекомендуется применять для всех каталогов со сценариями или для всех каталогов, где находятся файлы, используемые определенным кругом лиц, — например, администраторские сценарии;
- `Require` параметр — позволяет задать пользователей, которым разрешен доступ к каталогу. В качестве параметра можно указывать:
 - `user` — имена пользователей (или их ID), которым разрешен доступ к каталогу. Например: `Require user robert FlenovM`;
 - `group` — названия групп, пользователям которых позволен доступ к каталогу. Директива работает так же, как и `user`;
 - `valid-user` — доступ к каталогу разрешен любому аутентифицированному пользователю;
- `Satisfy` параметр — если указать значение `any`, то для ограничения доступа используется или логин/пароль, или IP-адрес. Для идентификации пользователя по двум условиям одновременно надо задать `all`;
- `AllowOverride` параметр — определяет, какие директивы из файла `.htaccess` в указанном каталоге могут перекрывать конфигурацию сервера. В качестве параметра можно указать одно из следующих значений: `None`, `All`, `AuthConfig`, `FileInfo`, `Indexes`, `Limit` и `Options`;
- `AuthName` — домен авторизации, который должен использовать клиент при определении имени и пароля;
- `Options` параметры — определяет возможности веб-сервера, которые доступны в указанном каталоге. Если у вас есть каталог, в который пользователям разрешено закачивать картинки, то вполне логичным является запрет на выполнение в нем любых сценариев. Не надо надеяться, что вы сумеете программно запретить загрузку любых файлов, кроме изображений. С помощью опций можно сделать так, чтобы сценарий веб-сервером не выполнялся.

В качестве аргумента параметры указывается `All` — означает разрешение всех опций, кроме `MultiViews`, `None` — запрещает все опции, или комбинация из следующих значений:

- `ExecCGI` — разрешено выполнение CGI-сценариев. Чаще всего для этого используется отдельный каталог `/cgi-bin`, но и в нем можно определить отдельные подкаталоги, в которых запрещено выполнение;
- `FollowSymLinks` — позволяет использовать символьные ссылки. Убедитесь, что в каталоге нет опасных ссылок и их права не избыточны. Мы уже говорили в *разд. 3.1.3*, что ссылки сами по себе опасны, поэтому с ними нужно обращаться аккуратно, где бы они ни находились;
- `SymLinksIfOwnerMatch` — следовать по символьным ссылкам, только если владельцы целевого файла и ссылки совпадают. При использовании символьных

ссылок в том или ином каталоге лучше указать этот параметр вместо `FollowSymLinks`. Если хакер сможет создать ссылку на каталог `/etc` и проследует по ней из веб-браузера, то это вызовет серьезные проблемы в безопасности;

- `Includes` — использовать SSI (Server Side Includes, включения на стороне сервера);
- `IncludesNOEXEC` — использовать SSI, кроме команд `exec` и `include`. Если вы не используете в сценариях CGI такие команды, то эта опция предпочтительнее предыдущей;
- `Indexes` — отобразить список содержимого каталога, если отсутствует файл по умолчанию. Чаще всего пользователи набирают адреса в укороченном формате — например: **www.cydsoft.com**. Здесь не указан файл, который нужно загрузить. Полный URL выглядит так: **www.cydsoft.com/index.htm**, но если указан только домен, сервер сам ищет файл по умолчанию и открывает его. Это могут быть файлы `index.htm`, `index.html`, `index.asp` или `index.php`, `default.htm` и т. д. Если ни один из таких файлов по указанному пути не найден, то при включенной опции `Indexes` будет выведен список содержимого каталога, а иначе — страница ошибки. Я рекомендую отключать эту опцию, потому что злоумышленник может получить слишком много информации о структуре каталога и его содержимом;
- `MultiViews` — представление зависит от предпочтений клиента.

Перед каждой из этих опций можно ставить знаки «плюс» или «минус», что соответствует разрешению или запрещению опции. Если знак не указан, то это соответствует разрешению.

Все описанные здесь директивы могут находиться не только в основном файле, но и в файлах `.htaccess`, располагаемых в отдельных каталогах для определения прав этих каталогов.

Права доступа можно определять не только для каталогов, но и для отдельных файлов. Это описание делается между двумя следующими строками:

```
<Files ИмяФайла>
</Files>
```

Такое объявление может находиться внутри объявления прав доступа к каталогу, например:

```
<Directory /var/www/html>
    Order allow,deny
    Allow from all
    <Files "/var/www/html/admin.php">
        Deny from all
    </Files>
</Directory>
```

Директивы для файла совпадают с директивами для каталогов. Поэтому в приведенном примере к подкаталогу `/var/www/html` разрешен доступ всем пользователям, а к файлу `/var/www/html/admin.php` из этого каталога запрещен доступ абсолютно всем, что, разумеется, не имеет особого смысла и приведено только для примера, поскольку в таком случае разумнее просто удалить этот файл, разве что запрет носит временный характер.

Помимо файлов и каталогов, можно ограничивать и методы протокола HTTP, такие как GET, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE, PATCH, PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK. Где тут собака зарыта? Допустим, что у вас есть сценарий, которому пользователь должен передать параметры. Это делается одним из методов: POST или GET. Если вы заведомо знаете, что программист использует только метод GET, то запретите все остальные, чтобы хакер не смог воспользоваться потенциальной уязвимостью в сценарии, изменив метод.

Но если честно, настраивать такое может быть накладно, поэтому никто этого не делает. Определить, с помощью какого метода допустимо обращаться к точке доступа, можно и средствами языка программирования.

Бывают ситуации, когда не всем пользователям позволено отправлять данные на сервер. Например, сценарии в определенном каталоге могут быть доступны для исполнения всем, но загружать информацию на сервер могут только администраторы. Эта проблема легко решается с помощью разграничения прав использования методов протокола HTTP.

Права на использование методов описываются следующим образом:

```
<Limit ИмяМетода>
    Права
</Limit>
```

Как видите, это похоже на описание разрешений на файлы, даже права доступа используются те же самые. Они размещаются внутри определения каталогов (`<Directory>` или `<Location>`) и влияют только на указанный каталог.

К примеру, так можно запретить любую передачу данных из каталога `/home` в любом направлении:

```
<Directory /home>
    <Limit GET POST>
        Deny from all
    </Limit>
</Directory>
```

При этом внутри определения прав для каталога `/home` устанавливается запрет на методы GET и POST.

Я всегда сначала закрываю все, что только можно, и только потом постепенно смягчаю права, чтобы все сценарии начали работать верно. Лучше лишний раз прописать явный запрет, чем потом упустить опасное разрешение.

7.4. Создание виртуальных веб-серверов

На одном физическом сервере может работать большое количество виртуальных веб-серверов — например: **www.your_name.com** и **www.your_company.com**. Это два разных веб-сайта, но они находятся на одном сервере. Такое расположение дает нам следующие преимущества:

- экономия денег на закупке серверов;
- эффективное использование каналов связи, если сайты небольшие и нагрузка на сервер невысока;
- экономия IP-адресов, лимит которых уже давно был бы исчерпан, если бы все сайты находились на выделенных серверах. Виртуальные веб-серверы могут иметь как отдельные IP-адреса, так и использовать общий адрес, а различаться доменными именами;
- упрощение администрирования и контроля за безопасностью. Конфигурация веб-сервера и его защита — достаточно сложный процесс, поэтому намного легче настроить и обновлять программное обеспечение одного физического сервера, чем сотни серверов, ресурсы каждого из которых используются на 1%.

С другой стороны, выделенные IP-адреса дают немного больше преимуществ по работе с сайтом, да и поисковые оптимизаторы любят именно выделенные адреса. Но это уже отдельная история.

Веб-сайты описываются в каталоге `/etc/apache2/sites-available`. Здесь уже может находиться файл сайта по умолчанию, а вы можете создать новые файлы. Например, для наших тестов можно создать файл `testsite.com.conf`. Теперь, чтобы получить доступ к этому сайту на компьютере, с которого вы будете загружать сайт, добавьте в файл `hosts` имя сайта и его адрес.

Допустим, вы хотите с этого же компьютера иметь доступ к своему сайту. Добавьте в свой файл `/etc/hosts` строку:

```
127.0.0.1 testsite.com
```

Адрес `127.0.0.1` всегда указывает на тот же компьютер, с которого производится подключение, т. е. на наш же сервер.

Из командной строки добавить что-то в файл `hosts` можно следующим образом:

```
echo 127.0.0.1 testsite.com | sudo tee -a /etc/hosts
```

Слева от вертикальной черты находится команда `echo`, которая выводит в консоль нужную нам строку. Команда справа от вертикальной черты читает данные команды слева. Тут у нас программа `tee`, которая все полученные данные записывает в файл. Поскольку файл `/etc/hosts` защищенный, мы пишем из-под `sudo`, а ключ `-a` позволяет добавлять в конец файла.

Теперь посмотрим, как в созданном файле `/etc/apache2/sites-available/testsite.com.conf` создать новый веб-сайт. Для этого откройте этот файл с помощью текстового редактора и записывайте в него весь приводимый далее текст.

Для создания виртуального сервера служит формат:

```
<VirtualHost адрес:порт>
</VirtualHost>
```

Между этими тегами указываются параметры виртуального сервера. Вот пример описания сервера, адрес которого 192.168.1.1 и порт 80:

```
<VirtualHost 192.168.1.1:80>
    ServerAdmin admin@your_server.com
    DocumentRoot /var/www/your_server
    ServerName your_server.com
    ErrorLog logs/your_server.com -error_log
    CustomLog logs/your_server.com -access_log common

    <Directory /var/www/your_server/>
        AllowOverride none
    </Directory>
</VirtualHost>
```

Рассмотрим только основные параметры, которые указываются при описании виртуального сервера:

- ☐ `ServerAdmin` — E-mail администратора, которому будут направляться сообщения об ошибках;
- ☐ `DocumentRoot` — каталог, в котором расположен корневой каталог сайта;
- ☐ `ServerName` — имя сайта. Если оно не указано, то используется локальный IP-адрес сервера.

Директивы `ErrorLog` и `CustomLog` нам уже знакомы. После этого в нашем примере идет указание прав доступа на каталог `/var/www/your_server/`, который является корнем для виртуального веб-сервера. Разрешения можно устанавливать как внутри объявления виртуального сервера, так и вне его.

Когда вы описали все необходимые параметры сайта, его нужно активировать, выполнив команду:

```
sudo a2ensite testsite.com.conf
```

Если у вас на сервере только пара веб-сайтов или вы работаете локально на своем компьютере, то можно держать всю конфигурацию в одном файле `/etc/apache2/extra/httpd-vhosts.conf`. В этом случае создавать файл `testsite.com.conf` надобности нет, а все, что мы описали ранее, просто помещаем в файл `httpd-vhosts.conf`.

Я веду на Youtube канал **Програмысли**, и там есть видео <https://youtu.be/GsoU4agdrUQ>, в котором я наглядно показываю, как настроить веб-сайт с использованием файла `httpd-vhosts.conf`.

За более подробной информацией обратитесь к документации по Apache.

7.5. Еще несколько слов о безопасности

В конфигурационном файле есть несколько директив, которые позволяют управлять безопасностью. Эти же команды можно указывать в файле `.htaccess`. Давайте их рассмотрим:

- ☐ `AuthType` параметр — тип аутентификации. В качестве параметра можно использовать одно из значений: `Basic` или `Digest`;
- ☐ `AuthGroupFile` параметр — файл, в котором хранится список групп пользователей;
- ☐ `AuthUserFile` параметр — файл, содержащий имена пользователей и пароли. Этот список лучше формировать утилитой `htpasswd`;
- ☐ `AuthAuthoritative` параметр — способ проверки прав. По умолчанию директива включена (`on`). Если она выключена (`off`), а пользователь не указал имя, то его аутентификация осуществляется другими модулями — например, по IP-адресу;
- ☐ `AuthDBMGroupFile` и `AuthDBMUserFile` — аналогичны `AuthGroupFile` и `AuthUserFile`, но в качестве параметра указывается файл в формате базы данных `Berkley-DB`.

Эти команды помогут вам настроить идентификацию пользователей при обращении к определенным каталогам. Их можно указывать как в основном файле сайта, так и в файле `.htaccess`.

Например, если у вас есть каталог, работа с которым разрешена только авторизованным пользователям, то можно указать файл паролей и директиву `Require valid-user`. Тогда при запросе файлов из этого каталога сервер предложит авторизоваться, указав имя пользователя и пароль.

7.5.1. Файлы `.htaccess`

Если какой-либо каталог веб-сервера должен иметь особые права, то лучше создать в этом каталоге файл `.htaccess`. Разрешения, описанные в таком файле, действуют на каталог, в котором он находится. Рассмотрим пример содержимого файла `.htaccess`:

```
AuthType Basic
AuthName "By Invitation Only"
AuthUserFile /pub/home/flenov/passwd
Require valid-user
```

В этом файле для текущего каталога указывается тип аутентификации `Basic`. Это значит, что будет использоваться окно для ввода имени и пароля. Текст, указанный в директиве `AuthName`, отобразится в заголовке окна. На рис. 7.1 приведен пример такого окна.

Директива `AuthUserFile` задает файл, в котором находится база имен и паролей пользователей сайта. Этот файл можно (и нужно) размещать вне области, доступной через веб-сервер. Последняя команда `Require` в качестве параметра использует значение `valid-user`. Это значит, что файлы в текущем каталоге смогут открыть только те пользователи, которые прошли аутентификацию.

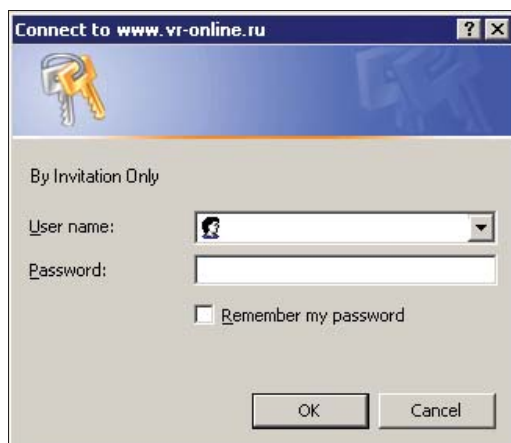


Рис. 7.1. Окно запроса имени и пароля

Вот таким простым способом можно запретить неавторизованный доступ к каталогу, содержащему секретные данные или сценарии администратора.

Как я уже говорил, в файле `.htaccess` могут находиться и директивы типа `Allow from`, которые мы рассматривали ранее (см. *разд. 7.3*).

Например, если нужно разрешить доступ только с определенного IP-адреса, то в файле может содержаться следующая строка:

```
Allow from 101.12.41.148
```

Если объединить защиту директивой `Allow from` и требование ввести пароль, то задача по взлому сервера сильно усложнится. Здесь уже недостаточно знания пароля — необходимо узнать конкретный IP-адрес для обращения к содержимому каталога, а это требует значительных усилий.

Эти же параметры можно указывать и в файле `httpd.conf`, например:

```
<directory /var/www/flenov/secret>
AuthType Basic
AuthName "By Invitation Only"
AuthUserFile /pub/home/flenov/passwd
Require valid-user
</directory>
```

Чем станете пользоваться вы, зависит от личных предпочтений. Мне больше нравится работать с файлом `.htaccess`, потому что настройки безопасности будут храниться в том же каталоге, на который устанавливаются права. Однако это небезопасно, потому что хакер может получить возможность прочитать этот файл, что не есть хорошо.

Использование централизованного файла `httpd.conf` дает преимущества, т. к. он находится в каталоге `/etc`, который не входит в корень веб-сервера и должен быть запрещен для просмотра пользователям.

7.5.2. Файлы паролей

Теперь нам предстоит узнать, как создаются файлы паролей и как ими управлять. Директива `AuthUserFile` для хранения информации об авторизации использует простой текстовый файл, в котором содержатся строки следующего вида:

```
flenov:{SHA}1ZZEBtPy4/gdHsyztjUEWb0d90E=
```

В этой записи два параметра, разделенные двоеточием. Сначала указано имя пользователя, а после разделителя — зашифрованный по алгоритму MD5 пароль. Формировать такой файл вручную сложно и не имеет смысла, поскольку для облегчения жизни администраторов есть утилита `htpasswd`. С ее помощью создаются и обновляются имена и пароли для базовой аутентификации веб-сервером HTTP-пользователей.

Удобство утилиты в том, что она может шифровать пароли и по алгоритму MD5, и с помощью системной функции `crypt()`. В одном файле могут находиться записи с обоими типами паролей.

Если вы храните имена и пароли в формате базы данных DBM (для ее указания в файле `.htaccess` служит директива `AuthDBMUserFile`), то для управления ею нужно применять команду `dbmmanage`.

Давайте рассмотрим, как пользоваться программой `htpasswd`. Общий вид команды ее вызова выглядит следующим образом:

```
htpasswd ключ файл_имя пароль
```

Пароль и имя файла являются необязательными, и их наличие зависит от указанных опций. Основные ключи программы:

- ❑ `-c` — создать новый файл. Если указанный файл уже существует, то он перезаписывается, и старое содержимое теряется. Например, можно написать:

```
htpasswd -c mypasswd robert
```

После выполнения этой команды перед вами появится приглашение ввести пароль для нового пользователя `robert` и подтвердить его. В результате будет создан файл `mypasswd`, содержащий одну запись для пользователя `robert` с указанным вами паролем;

- ❑ `-m` — использовать модифицированный Apache алгоритм MD5 для паролей. Это позволит переносить созданный файл на любую другую платформу (Windows, UNIX, BeOS и т. д.), где работает веб-сервер Apache. Такой ключ удобен, если у вас разнородная сеть и один файл с паролями используется на разных серверах;
- ❑ `-d` — применить системную функцию `crypt()` для шифрования;
- ❑ `-s` — применить SHA-шифрование (на базе алгоритма хеширования), которое используется на платформе Netscape;
- ❑ `-p` — не шифровать пароли. Я не рекомендую использовать этот ключ, потому что он небезопасен;
- ❑ `-n` — не вносить никаких изменений, а только вывести результат на экран.

Для добавления нового пользователя можно выполнить команду без указания ключей, а передать в качестве параметров только имена файла и пользователя:

```
htpasswd .htaccess Flenov
```

Надо отметить, что хотя пароль можно задавать в командной строке, делать это не следует, поскольку такая команда небезопасна. Команда может сохраниться в истории командной оболочки, и незашифрованный пароль таким (или другим) образом окажется известным хакеру. Если пароль не задан, но требуется, то программа запросит его в интерактивном режиме.

У команды `htpasswd` есть два ограничения: в имени пользователя не должно быть символа двоеточия, и пароль не может превышать 255 символов. Оба условия достаточно демократичны, и с ними можно смириться. Из моих знакомых столь длинный пароль пока еще никто не захотел устанавливать, а использовать имя пользователя с двоеточием редко кому приходит на ум.

7.5.3. Проблемы авторизации

Авторизация — это слишком простой способ обеспечения безопасности. При передаче пароли шифруются простым кодированием Base64. Если хакер сможет перехватить пакет, содержащий имя пользователя и пароль, то он прочитает эту информацию через пять секунд. Для расшифровки Base64 не нужно подбирать пароль, достаточно выполнить одну функцию, которая вскроет информацию практически моментально.

Для создания реально безопасного соединения необходимо сначала зашифровать весь трафик с помощью протокола HTTPS, который использует SSL. О нем мы еще поговорим в *разд. 7.9*.

7.5.4. Обработка на сервере

HTML-файлы могут обрабатываться прямо на сервере (так же, как выполняются файлы PHP). С одной стороны, это удобно, потому что код PHP можно будет вставлять в файлы с расширением `htm`; с другой стороны, HTML-файлы далеко небезопасны, и если хакер сможет их изменять, то сервер окажется под угрозой.

Чтобы разрешить серверу выполнять файлы с определенными расширениями, служит директива `AddHandler`. В конфигурационном файле `httpd.conf` можно найти следующие строки с этой командой:

```
AddHandler cgi-script .cgi
AddHandler server-parsed .shtml
```

Если у вас нет необходимости использовать интерпретатор языка Perl, то первую строку следует закомментировать, чтобы она даже не смущала. Вторая строка безобидна, но если таким же образом разрешить серверу работать с HTML- или HTML-файлами, то это уже станет небезопасным. Следующей строки в вашем конфигурационном файле быть не должно:

```
AddHandler server-parsed .html
```

Если где-то действительно есть необходимость выполнять включения в HTML-документах, то пропишите это в файле `.htaccess`. В остальных каталогах я рекомендую в явном виде запретить обработку HTML-файлов сервером. Для этого добавьте следующую строку в конфигурационный файл `httpd.conf` или в файл `.htaccess` каждого каталога:

```
RemoveHandler .html .htm
```

Так мы запретим выполнение файла на сервере, но не отменим SSI-инструкции. Например, следующий код в SHTML-файле будет выполнен:

```
<!--#include virtual="filename.shtml" -->
```

Если же вы не используете SSI и, соответственно, SHTML-файлы, то закомментируйте и следующую строку (по умолчанию она активна):

```
AddHandler server-parsed .shtml
```

7.6. Проще, удобнее, быстрее

Процесс конфигурирования должен быть максимально удобным. До появления Apache2 все настройки находились в одном файле `/etc/httpd/conf/httpd.conf`, и разобраться в них было очень сложно. А чем больше параметров, тем выше вероятность, что вы что-либо пропустите. Сейчас конфигурация разбита на несколько файлов, но все равно они достаточно большие. Чтобы упростить поддержку вашего веб-сервера, могу посоветовать придерживаться следующих рекомендаций:

- для удобства администрирования все описания прав доступа можно перенести в конфигурационный файл `.htaccess`. Выделение части разрешений в отдельный файл действительно может помочь в управлении, потому что права в отдельном файле будут видны как на ладони;
- комментируйте все ваши действия. Многие настройки не изменяются годами, и уже через полгода трудно вспомнить, зачем вы установили определенную директиву. Например, вы запретили доступ для всех пользователей к какому-либо каталогу, который временно использовался для тестирования сценариев. Через какое-то время вы можете забыть об этом и случайно открыть доступ к сценариям, которые не были полностью отлажены и могут стать причиной взлома или крушения системы.

Если вы последовали первым двум рекомендациям, то нужно включить эти файлы `.htaccess` в основной файл при помощи директивы `Include`:

```
Include /etc/httpd/conf/access.conf
```

При этом стоит проверить, что такая директива не включена в файл по умолчанию.

Чем удобнее управлять безопасностью сервера, тем меньше ошибок вы совершите. Если все параметры хорошо видны, а подробные комментарии напоминают вам о назначении сделанных настроек, управление будет удобнее. Такой подход к администрированию поможет и оперативно решать возникающие проблемы. А, как

известно, в войне администраторов и хакеров побеждает тот, кто больше знает, имеет больше опыта и быстрее реагирует. Последнее очень важно.

Централизованное хранение прав доступа в конфигурационных файлах веб-сервера приемлемо только на небольших сайтах. Когда работает сотня виртуальных серверов, то такие описания становятся слишком громоздкими.

Для больших сайтов я рекомендую описывать в конфигурационных файлах сервера только общие правила, которые затрагивают сразу несколько каталогов. Это возможно, потому что при указании пути к каталогу можно использовать регулярные выражения. Приведу пример, который определяет правила для всего, что находится в каталоге `/home`:

```
<Directory /home/* >
  AllowOverride FileInfo AuthConfig Limit
  Options MultiViews Indexes SymLinksIfOwnerMatch IncludesNoExec
  <Limit GET POST OPTIONS PROPFIND>
    Order allow,deny
    Allow from all
  </Limit>
  <LimitExcept GET POST OPTIONS PROPFIND>
    Order deny,allow
    Deny from all
  </LimitExcept>
</Directory>
```

С помощью таких регулярных выражений удобно создавать общие правила для различных каталогов. Например, если указать в качестве каталога значение `/home/*/public_html`, то все каталоги `public_html` в каталоге `/home` и любом его подкаталоге будут иметь указанные права, если они явно не переопределены для отдельных папок. Это значит, что под шаблон попадут такие папки, как `/home/robert/public_html`, `/home/other_user/public_html` и т. п.

7.7. Безопасность сценариев

Как мы уже говорили, сценарии представляют для веб-сервера серьезную опасность. Через ошибки в них происходило очень много громких взломов. Мы уже знаем, что необходимо отключить все интерпретаторы, которые не используются, и оставить только те, что нужны. Этим мы усложним задачу хакера, но не решим проблему полностью.

Наиболее безопасный сайт — это тот, что использует статичные (HTML) документы и не имеет сценариев, выполняемых на сервере (PHP, ASP, Perl, Python и др.). Правда, такой сайт будет практически бесполезен и никому не нужен. Поэтому возможности того интерпретатора, который вы все же используете, необходимо максимально ограничить.

Допустим, ваш сайт использует сценарии PHP и в этих сценариях есть функции, которые обращаются к системе. Если вы их неверно пропишете (например, не ста-

нут проверяться параметры, заданные пользователем), то злоумышленник получит возможность передать такие значения, которые могут нарушить работу сервера. Мы не будем говорить о том, как правильно писать сценарии и как их делать безопасными, потому что это задача программистов. Но не стоит надеяться на их профессионализм, потому что все мы — люди и нам свойственно ошибаться. Администратор должен сделать все, чтобы погрешности сценариев не стали фатальными.

В интерпретаторе PHP есть возможность описывать правила выполнения каких-либо действий, используя более безопасные настройки и права доступа, — это режим `safe_mode`. Но вы должны отдавать себе отчет в том, что некоторые скрипты могут отказаться работать в этом режиме. Если администраторы отключают `safe_mode`, то это не совсем верно. Я всегда сначала проверяю, можно ли переписать сценарий так, чтобы можно было использовать этот режим, и выключаю безопасный режим, лишь если это нереально.

При этом следует учесть, что параметр `safe_mode` отключает те функции, которые большинству сайтов действительно не нужны. Так что лучше все же оставить параметр включенным и подправить/заменить скрипты.

Кстати, все настройки для интерпретатора PHP хранятся в файле `/etc/php.ini`. Мы его рассматривать не будем, потому что эта тема выходит за рамки книги про Linux. Еще один намек на хороший параметр — магические кавычки: `magic_quotes`.

7.7.1. Основы безопасности сценариев

В настоящее время большинство взломов в Интернете совершается с помощью или благодаря ошибкам в сценариях веб-страниц. Давайте попробуем разобраться, почему это происходит.

Большинство владельцев домашних сайтов — это простые пользователи, которые хотят быстро получить собственную страницу с множеством возможностей. А что именно нужно сайту? Конечно же, это гостевая книга, форум, чат, голосование и т. д. Все эти разделы нельзя сделать с помощью языка разметки HTML, и нужны знания программирования — например, на Perl или PHP. Владельцы домашних сайтов не хотят (или не могут) вникать в тонкости программирования, поэтому используют в своих проектах уже готовые (платные или бесплатные) движки.

Как я уже неоднократно говорил, любая разработка может содержать ошибки, просто о них до поры до времени никому не известно. Если вы в каком-то журнале прочитали, что веб-движок XXX супербезопасен, то не стоит сразу же этому верить. Журналисты часто используют громкие утверждения, чтобы сделать статью красивее. Но реальность такая красивая не всегда.

Распространенная программа становится лакомым кусочком для любого хакера, потому что ее взлом позволяет приникнуть сразу во множество систем, где она установлена.

Если администратор сайта устанавливает на свою страничку популярный форум, то он должен понимать, что в нем когда-нибудь найдут уязвимость и через нее какой-

нибудь злоумышленник проникнет в систему. Чтобы этого не произошло, администратор сайта должен регулярно обновлять используемые веб-программы и веб-сценарии.

Если вы знаете хотя бы основные принципы защиты веб-приложений и решили написать собственный форум для сайта, то ваша работа может оказаться безопаснее, чем при использовании любой готовой программы, и тем более программы с открытым кодом, по следующим причинам:

- ☐ я надеюсь, вы будете реализовывать только те функции, которые действительно вам нужны. А чем меньше функций, тем меньше вероятность ошибки. Универсальные скрипты иногда слишком сложные и пытаются объять как можно больше, что может стать причиной ошибки;
- ☐ ваш исходный код будет недоступен. Это не решает проблемы безопасности, а только немного усложняет работу взломщика;
- ☐ ценность взлома вашего скрипта не так высока, как взлом чего-то очень популярного.

Ну а если вы не знаете особенностей языка или вообще не знакомы с программированием, то лучше за это и не браться, — даже начинающий хакер найдет в вашем продукте уязвимость, пусть и не зная исходного кода, структуры базы данных и других параметров, упрощающих взлом веб-сценария.

Если вы отвечаете за один веб-сервер, то легко сможете контролировать обновление программ. Хуже всего администраторам хостинговых компаний. На их серверах расположены сотни, а то и тысячи веб-сайтов. Проследить за всеми хозяевами сайтов нереально, поэтому нужно защитить свои владения от недобросовестных или ленивых пользователей. В этих целях лучше всего использовать программу jail, о которой мы говорили в *главе 4*. Для веб-сервиса вы должны создать его собственный виртуальный сервер, в котором он и будет работать. Если злоумышленник взломает систему через веб-программу, то сможет нарушить работу только виртуального каталога.

Во время подготовки материалов для предыдущего издания этой книги в одном из популярных форумов была найдена уязвимость, позволяющая выполнять на удаленной системе любые команды. Для этого нужно было специальным образом передать директиву через строку URL. Эту главу к тому изданию я начал писать через месяц после этого страшного открытия и как-то раз, вспомнив про найденную в форуме ошибку, решил проверить в Интернете все сайты, содержащие уязвимый форум. Я запустил по серверам соответствующий поиск, и — вы не поверите, но их было сотни.

Меня заинтересовала пара сайтов, расположенных на серверах крупных хостинговых компаний. На обоих я выполнил команду `ls -a /etc`. Результат не заставил себя ждать — я увидел весь каталог `/etc`, и моих прав хватало даже на удаление файлов из папки, где находятся файлы веб-сайта. Конечно же, делать этого я не стал даже в тестовых целях. Для доказательства серьезности ошибки я переименовал по одному файлу на каждой системе и сообщил об уязвимости их администраторам.

ВНИМАНИЕ!

Не советую вам повторять подобные действия. Взлом даже с добрыми намерениями не всеми администраторами воспринимается положительно. Некоторые могут сообщить о ваших действиях в правоохранительные органы, и тогда вам не избежать судебных тяжб. Благие побуждения могут быть восприняты неверно. Если я нахожу какую-либо дыру, то всегда сообщаю администраторам, но отправляю для этого письмо анонимно.

Это было давно, сейчас система уже в конфигурации «из коробки» защищает все файлы каталога `/etc`, и никаких дополнительных настроек делать не требуется.

Переместив Apache в виртуальное пространство, вы обезопасите только свою систему, но все сайты, которые работают в этом пространстве, остаются уязвимыми. Здесь уже нужно искать другие методы защиты — например, через права доступа, запуск нескольких копий Apache (каждая работает в своем пространстве), выделенные серверы для каждого сайта, запрет на выполнение опасных функций в интерпретаторе PHP и т. д.

Особо важные сайты должны располагаться на выделенном сервере и находиться под пристальным присмотром. Например, нельзя размещать на одном физическом сервере сайт электронного магазина и домашние странички, очень часто использующие бесплатные или некачественные модули, в которых бывают ошибки и пользователи которых к тому же не обновляют эти программы. Когда-нибудь злоумышленник взломает домашнюю страничку через ее уязвимые сценарии и сможет украсть номера кредитных карт пользователей интернет-магазина. Это поставит крест на вашей карьере администратора.

7.7.2. Модуль `mod_security`

Несмотря на то что безопасность веб-сервера в основном зависит от выполняемых на нем скриптов-сценариев и программистов, которые их пишут, есть возможность защитить сервер вне зависимости от этих факторов. Отличное решение проблемы — бесплатный модуль для Apache под названием `mod_security`.

Принцип действия этого модуля схож с сетевым экраном, который встроен в ОС, только в данном случае он специально разработан для обеспечения взаимодействия по протоколу HTTP. Модуль на основе правил, которые задает администратор, анализирует запросы пользователей к серверу и выносит свое решение о возможности пропустить пакет к веб-серверу.

Правила определяют, что может быть в запросе, а что нет. Там обычно содержится URL-адрес, с которого необходимо взять документ или файл. Как можно сформулировать правило для модуля с точки зрения безопасности системы?

Рассмотрим простейший пример — для сервера опасно незаконное обращение к файлу `/etc/passwd`, а значит, его не должно быть в строке URL. Таким образом, сетевой экран проверяет на основе заданных фильтров адрес URL, и если он нарушает правила, то запрос отклоняется.

Модуль `mod_security` надо скачать с сайта www.modsecurity.org. После его установки в файле `httpd.conf` появляется возможность применять дополнительные директивы фильтрации запросов. Рассмотрим наиболее интересные из них:

- ❑ `SecFilterEngine On` — включить режим фильтрации запросов;
- ❑ `SecFilterCheckURLEncoding On` — проверять кодировку URL;
- ❑ `SecFilterForceByteRange 32 126` — использовать символы только из указанного диапазона. Существует достаточное количество служебных символов (например, перевод каретки, конец строки), коды которых менее 32. Большинство из них невидимы, но требуют обработки нажатия соответствующих клавиш. Но как же тогда хакер может ввести такой символ в URL? Только через их код. Например, чтобы поместить в URL символ перевода строки, необходимо указать в адресе `%13`. В этой директиве коды символов менее 32 и более 126 объявляются недопустимыми для адреса, и такие пакеты не пропускаются к веб-серверу;
- ❑ `SecAuditLog logs/audit_log` — определяет файл журнала, в котором будет храниться информация об аудите;
- ❑ `SecFilterDefaultAction "deny,log,status:406"` — задает действие по умолчанию. Сейчас здесь указан запрет (`deny`);
- ❑ `SecFilter xxx redirect:http://www.webkreator.com` — обеспечивает переадресацию. Если правила соблюдены, то пользователя отправляют на сайт <http://www.webkreator.com>;
- ❑ `SecFilter yyy log,exec:/home/apache/report-attack.pl` — запускает сценарий. Если фильтр срабатывает, то будет выполнен скрипт `/home/apache/report-attack.pl`;
- ❑ `SecFilter /etc/passwd` — устанавливает запрет на использование в запросе пользователя подстроки `/etc/passwd`. Таким же образом нужно добавить ограничение на файл `/etc/shadow`;
- ❑ `SecFilter /bin/ls` — отказ пользователям в обращении к директивам. Здесь запрещается команда `ls`, которая может позволить хакеру увидеть содержимое каталогов, если в сценарии есть ошибка. Необходимо предотвратить обращения к таким командам, как `cat`, `rm`, `cp`, `ftp` и др.;
- ❑ `SecFilter "\.\/"` — классическая атака, когда в URL указываются две точки подряд для перехода в родительский каталог. Их там быть не должно;
- ❑ `SecFilter "delete[:space:]+from"` — запрет текста `delete ... from`, который чаще всего содержится в SQL-запросах для удаления данных. Такая строка может использоваться в атаках типа SQL injection. Помимо этого, рекомендуется установить следующие три фильтра:
 - `SecFilter "insert[:space:]+into"` — используется в SQL-запросах для добавления данных;
 - `SecFilter "select.+from"` — используется в SQL-запросах для чтения данных из таблицы базы данных;

- `SecFilter "<(.\|\\n)+>"` и `SecFilter "<[[:space:]]*script"` — позволяет защититься от XSS-атак (Cross-Site Scripting, межсайтовое выполнение сценариев).

Вот основные методы, использование которых может повысить безопасность вашего веб-сервера, а также защищать целые сети из серверов. Дополнительную информацию можно получить с сайта разработчиков модуля `mod_security`.

7.7.3. Секреты и советы

Как бы аккуратно программист ни писал сценарии, как бы вы их ни защищали с помощью специальных модулей, неплохо предпринять дополнительные меры безопасности. Есть еще ряд параметров, которыми можно воспользоваться для этих целей. В этом разделе я собрал краткие рекомендации, которые помогут вам сделать необходимые настройки.

Ограничение сценариев

Во-первых, ограничьте выполнение сценариев только отдельным каталогом. Чаще всего это каталог `cgi-bin`.

Резервные копии

Никогда не сохраняйте резервные копии сценариев в каталогах, доступных веб-серверам. Например, если на сервере есть PHP-скрипты, то пользователи видят только результат их выполнения. Чтобы посмотреть исходный код, хакеру необходим доступ к серверу — например, FTP или Telnet, поскольку сервер Apache не передает таких данных клиенту.

Перед тем как вносить изменения в сценарий, программисты любят создавать на сервере их резервные копии, чтобы в случае ошибки можно было восстановить старую, но рабочую версию. Для этого очень часто содержимое файла копируется в тот же каталог, под тем же именем, но с другим расширением — например, `old` или `bak` (наиболее часто встречаемые).

Такие программы уже не выполняются сервером, но если хакер запросит такой файл, то он увидит его исходный код, что поможет ему быстрее найти ошибки в скриптах.

Когда злоумышленник исследует сценарии на сервере, то никто не мешает ему проверить наличие резервной копии. Увидев, что у вас есть файл `www.servername.com/index.php`, хакер попытается загрузить `www.servername.com/index.bak` или `www.servername.com/index.old`. На непрофессиональных сайтах такие версии встречаются очень часто. Не допускайте подобных промахов.

Любой специалист по безопасности должен запретить работу с резервными копиями. Сколько бы мы ни говорили программистам о том, что нельзя держать на сервере ничего лишнего, они все равно продолжают это делать, потому что им так удобно. Наша задача — сделать хранение этих копий безопасным, т. е. запретить к ним доступ со стороны веб-клиентов.

Как уже отмечалось, чаще всего для резервных копий файлов используются те же имена файлов, но расширение меняется на bak или old. Следующий пример запрещает пользователям доступ к таким файлам:

```
<FilesMatch "\.bak$">
  Order deny, allow
  Deny from all
</FilesMatch>

<FilesMatch "\.old$">
  Order deny, allow
  Deny from all
</FilesMatch>
```

7.8. Индексация веб-страниц

За последние годы Интернет разросся до таких размеров, что найти в нем что-либо без хорошей поисковой системы стало невозможным. Первые такие системы просто индексировали страницы по их содержимому и потом использовали полученную базу данных для поиска, что давало очень приблизительные результаты. Если ввести в качестве контекста слово *лук*, то будет отобрано огромное количество сайтов по пищевой промышленности и по стрельбе из лука. В большинстве языков есть слова, которые имеют несколько значений, и поиск по ним затруднителен.

Проблема не только в двусмысленности некоторых слов. Есть множество широко употребляемых выражений, по которым тоже сложно произвести точную выборку. В связи с этим поисковые системы стали развиваться, и теперь можно добавлять в запрос различные параметры. Одной из самых мощных является поисковая система Google (www.google.com). В ней реализовано много возможностей, позволяющих сделать поиск более точным. Жаль, что большинство пользователей не освоили их, а вот взломщики изучили все такие функции и используют в своих целях.

Один из самых простых способов взлома — найти с помощью поисковой системы закрытую веб-страницу. Некоторые сайты имеют засекреченные области, к которым доступ осуществляется по паролю. Сюда же относятся платные ресурсы, где защита основана на проверке пароля при входе, а не на защите каждой страницы и использовании SSL. При этом часто случается, что Google индексирует запрещенные страницы, и их можно просмотреть через поиск. Для этого всего лишь надо четко знать, какая информация хранится в файле, и как можно точнее составить строку поиска.

С помощью Google можно найти довольно важные данные, которые скрыты от пользователя, но по ошибке администратора стали доступными для индексирующей машины Google, — надо только во время поиска правильно задать параметры. Например, можно ввести в строку поиска следующий запрос:

```
Годовой отчет filetype:doc
```

Или

Годовой отчет filetype:xls

И вы найдете все документы в формате Word и Excel, содержащие слова «Годовой отчет». Возможно, документов будет найдено слишком много, поэтому запрос придется ужесточить, но кто ищет, тот всегда найдет. Существуют реальные примеры из жизни, когда таким простым способом были найдены секретные данные, в том числе действующие номера кредитных карт и финансовые отчеты фирм.

Мое мнение — очень глупо держать какие-то отчеты на открытых серверах. Уж если Google может добраться до них, то и другие смогут. Но почему-то такие файлы продолжают держать в открытую.

Давайте рассмотрим, как можно запретить индексацию каталогов веб-страниц, которые не должны стать доступными для всеобщего просмотра. Для этого необходимо понимать, что именно индексируют поисковые системы. На этот вопрос ответить легко — все, что попадает под руку: текст, описания, названия картинок, документы поддерживаемых форматов (PDF, XLSX, DOCX и т. д.).

Наша задача — ограничить настойчивость индексирующих роботов поисковых машин, чтобы они не трогали то, что запрещено. Для этого робот должен получить определенный сигнал. Как это сделать? Было найдено достаточно простое, но элегантное решение — в корень сайта помещается файл с именем robots.txt, который содержит правила для поисковых машин.

Допустим, что у вас есть сайт **www.your_name.com**. Робот, прежде чем начать свою работу, пробует загрузить файл **www.your_name.com/robots.txt**. Если он будет найден, то индексация пойдет в соответствии с описанными в файле правилами, иначе процесс затронет все подряд.

Формат файла очень прост и состоит всего лишь из двух директив:

- ☐ **User-Agent:** параметр — в качестве параметра передается имя поисковой системы, к которой относятся запреты. Таких записей в файле может быть несколько, и каждая будет описывать свою поисковую систему. Если запреты должны действовать на все поисковики, то достаточно указать в начале файла директиву **User-Agent** с параметром звездочка (*);
- ☐ **Disallow:** адрес — запрещает индексировать определенный адрес, который указывается относительно URL. Например, если вы хотите отказаться от индексации страниц с URL **www.your_name.com/admin**, то в качестве параметра нужно указать **/admin**. Как видите, этот адрес берется именно из URL, а не из вашей реальной файловой системы, потому что поисковая система не может знать истинное положение файлов на диске сервера и оперирует только адресами URL.

Вот пример файла robots.txt, который запрещает индексацию страниц, находящихся по адресам **www.your_name.com/admin** и **www.your_name.com/cgi_bin**, для любых индексирующих роботов поисковых систем:

```
User-Agent: *
Disallow: /cgi-bin/
Disallow: /admin/
```

Эти правила запрещают индексацию с учетом подкаталогов. Например, файлы по адресу **www.your_name.com/cgi_bin/forum** тоже не будут индексироваться.

Следующий пример запрещает индексацию сайта вовсе:

```
User-Agent: *  
Disallow: /
```

Если на вашем сайте есть каталог с секретными данными, то следует запретить их индексацию. Но это не является эффективным методом защиты, потому что файл `robots.txt` может скачать любой желающий, а, значит, это даже лишний раз укажет, где искать важные файлы и документы. Так что лучший метод защиты — не хранить ничего важного на открытых серверах. Лучше для этого создавать отдельные сайты, которые будут доступны только с определенных IP-адресов и только с доступом по паролю.

7.9. Безопасность подключения

Различные технологии прослушивания сетевого трафика в основном эффективны в локальных сетях, но хакеры больше любят интернет-соединения, потому что здесь можно найти больше интересного и есть лазейка, чтобы удаленно проводить атаку.

Что опасного может увидеть хакер, когда клиент просматривает страницы на веб-сервере? Мы каждый день вводим на веб-страницах какие-либо данные, пароли, номера кредитных карт, и именно это является основной целью хакера.

Как можно, например, находясь в Европе, перехватить трафик, который проходит между двумя городами в США? Скорее всего, пакеты будут следовать по каналам США, и в Европе им делать нечего. Но задача хакера сделать свой компьютер посредником в передаче пакетов данных — чем-то наподобие прокси-сервера.

Самое сложное — организовать, чтобы клиент подключился не к реальному веб-серверу, а к вашему компьютеру. Чаще всего мы в браузерах набираем символьные имена адресов, но соединение происходит по IP-адресу. Для такого сопоставления используются DNS-серверы. Хакер может обмануть клиента с помощью ложного DNS-ответа или подставного DNS-сервера и тем самым перенаправить трафик на себя.

Затем компьютер злоумышленника будет переадресовывать пакеты реальному веб-серверу и возвращать ответы клиенту (рис. 7.2). Таким образом, весь трафик станет проходить через компьютер хакера.



Рис. 7.2. Перехват трафика

Но этот метод был хорош несколько лет назад, когда не было HTTPS-протокола и безопасного соединения с помощью SSL-шифрования. Тогда было проще обмануть серверы DNS.

Давайте вспомним, что для подключения по SSL программа-клиент и сервер обмениваются ключами, с помощью которых происходит шифрование. Для HTTPS помимо открытого и закрытого ключей необходимы подтвержденные сертификаты, которые выдаются специализированными центрами и подтверждают подлинность ключа. Программа-клиент проверяет сертификат, и если он достоверен (цифровая подпись принадлежит авторизованной фирме), то подключение разрешается. Сертификаты можно сгенерировать самостоятельно, а вот подпись подделать практически невозможно.

Если компьютер хакера будет просто передавать данные между клиентом и сервером, то трафик останется зашифрованным, и просмотреть его ему не удастся. Единственным вариантом может быть следующая схема:

1. На компьютере хакера генерируется пара ключей и сертификат.
2. Клиент подключается к компьютеру хакера и обменивается с ним ключами.
3. При передаче информации шифрование происходит с ключом, который сгенерировал хакер, поэтому он без проблем расшифровывает все данные.
4. Компьютер хакера соединяется с веб-сервером и получает его ключ.
5. Между компьютером хакера и веб-сервером устанавливается соединение по ключу веб-сервера.

При такой схеме клиент получает ключ, который сгенерирован хакером и не имеет нужной подписи. Это значит, что у клиента появится сообщение о подключении к сайту без необходимого сертификата. И вот тут происходит самое страшное — большинство пользователей, которые давно работают с Интернетом, устали смотреть на различные предупреждения, поэтому, не читая сообщения, нажимают кнопку **ОК** для продолжения работы. Даже я никогда не смотрю на строку в браузере с предупреждением о безопасном соединении.

Решить проблему подложного подключения можно только защитой DNS-сервера, чтобы хакер не смог стать посредником в соединении между клиентом и сервером. Не используйте прокси-серверы, в происхождении которых вы не уверены, ведь они могут принадлежать хакеру, и тогда весь ваш веб-трафик оказывается в опасности.

Единственное, что должен помнить пользователь: при неавторизованном соединении нельзя вводить действительно секретную информацию — например, номера кредитных карт. Вот это предупреждение должно появляться большими буквами при подключении к серверу с неподписанным сертификатом.

Когда мне нужно ввести где-то данные кредитной карты, вот тогда я обязательно смотрю на строку URL браузера, чтобы убедиться, что соединение установлено по HTTPS и что сертификат там подписан известным мне центром.



ГЛАВА 8

Электронная почта

Для кого-то Интернет — это просмотр сомнительных страниц на WWW, для некоторых — способ найти соперника или напарника в игре, а многие используют сеть для работы или обучения. Но все мы не можем жить без общения, и, несмотря на новые технологии, которые выдумывают для облегчения общения (социальные сети, мессенджеры и т. п.), электронная почта жила, живет и будет жить. Именно с электронной почты начали развиваться сети, и она была одним из первых сервисов Интернета.

Лично для меня почтовый клиент стал одной из основных программ, которой я пользуюсь каждый день. Я веду переписку с читателями, друзьями, начальством и т. д. Так уж получилось, что люди, с которыми я работаю, находятся в других городах и даже странах, и расстояние между нами измеряется тысячами километров. Если раньше это было сопряжено с большими проблемами, то теперь, благодаря компьютеру и Интернету, я могу, к примеру, жить в теплых краях, а выполнять работы для компании, которая находится на Аляске. Таким образом, легко и ноги сохранить в тепле, и зарабатывать деньги в холодных областях, хотя последние годы у меня получается наоборот — живу в Канаде, а работаю на более южную страну — США.

Как работает электронная почта (E-mail)? Рассмотрим основные моменты доставки письма:

1. Пользователь создает в почтовом клиенте (программе электронной почты) письмо, указывает получателя и отправляет его почтовому серверу. В настоящее время для передачи сообщений чаще всего используются SMTP-серверы, а для работы с ними — протокол SMTP.
2. Сервер, получив письмо, определяет место назначения, т. е. имя сервера. Адрес состоит из двух частей: имени пользователя и имени сервера, разделенных между собой знаком @, — например, **djon@servername.com**. Здесь **djon** — это имя пользователя, а **servername.com** — имя сервера. SMTP-сервер с помощью DNS узнает IP-адрес сервера (в нашем случае **servername.com**), которому должно быть доставлено письмо.

3. Письмо направляется серверу, на котором зарегистрирован получатель.
4. Получив письмо, сервер **servername.com** проверяет наличие адресата (у нас это пользователь **djon**), и если он присутствует, помещает письмо в его почтовый ящик. Если указанного пользователя нет, сервер сообщает об этом отправителю, возвращая письмо или его часть.
5. Пользователь просматривает свой почтовый ящик с помощью почтового клиента и может скачать письмо для чтения. Для чтения писем сейчас чаще всего используются протоколы POP3, MAPI или IMAP.

Описанный процесс похож на работу традиционной почты. В роли серверов там выступают почтовые отделения, которые сортируют почту и, в зависимости от адреса назначения, пересылают письма на почтовое отделение получателя.

Как я уже отметил, для передачи сообщений служит протокол SMTP, разработанный еще на заре становления Интернета. Его функций уже давно недостаточно, но он не утратил своей актуальности. Многие с удовольствием заменили бы этот протокол чем-нибудь более безопасным, но нынешние серверы слишком популярны, чтобы в один миг взять и отказаться от них.

Несколько десятков лет назад для работы с почтой широко использовался протокол UUCP (UNIX to UNIX Copy, копирование между UNIX-системами). Но он был слишком сильно привязан к ОС и при этом обладал ограниченными возможностями, поэтому не получил распространения и в настоящее время практически не применяется.

Для приема почты используются три протокола:

- ❑ POP3 (Post Office Protocol v3, почтовый протокол версии 3) — в настоящее время наиболее распространенный протокол приема почты;
- ❑ IMAP4 — существуют две интерпретации этого сокращения: Internet Message Access Protocol (Протокол доступа к сообщениям в сети Интернет) и Interactive Mail Access Protocol (Протокол интерактивного доступа к электронной почте). Этот протокол обладает большими возможностями по сравнению с POP3 — например, позволяет управлять сообщениями прямо на сервере, загружая на компьютер пользователя лишь заголовки писем для последующего отбора;
- ❑ MAPI (Messaging Application Programming Interface, интерфейс прикладного программирования электронной почты) — работает в сетях Microsoft на серверах Microsoft Exchange.

Кроме того, сейчас очень распространены базирующиеся в Сети системы просмотра почты, когда пользователи вовсе не скачивают почту на свой компьютер, а просматривают ее онлайн, используя браузер.

Самым распространенным средством доставки почты в Linux является весьма старая программа **sendmail**. Она обладает большими возможностями, но довольно сложна в реализации. Из-за почтенного возраста в сервере **sendmail** сохранилась и возможность передачи по протоколу UUCP, которая сейчас используется крайне редко.

Принцип работы sendmail достаточно прост. Получив письмо от клиента, программа определяет получателя и заносит необходимую для доставки служебную информацию в заголовок письма. Дальнейшие действия зависят от настроек. Например, письмо может быть отослано немедленно или помещено в хранилище. Через определенные промежутки времени накопившиеся письма отправляются своим адресатам.

8.1. Настройка sendmail

Основной конфигурационный файл, который вам понадобится, — `/etc/sendmail.cf` или `/etc/mail/sendmail.cf`. Если файла не существует, то, скорее всего, sendmail не установлен, что случается в установке дистрибутива по умолчанию. Устанавливаем:

```
sudo apt-get install sendmail
```

После установки сразу же должна начаться автоматическая настройка. Если и после этого файла конфигурации не оказалось в системе, то выполните эту команду:

```
sudo sendmailconfig
```

Программа начальной настройки запросит пути к конфигурационным файлам. Согласитесь со значениями по умолчанию, но обратите внимание, какие пути у вас спрашивают. В последнем релизе дистрибутива Ubuntu версия sendmail предлагает использовать путь `/etc/mail/` для всех файлов.

Сервер sendmail имеет плохую репутацию в связи со сложностью его настройки. Действительно, стоит лишь взглянуть на размер файла `sendmail.cf`, и становится жутко — это 32 килобайта (более 1000 строк). А если заглянуть внутрь файла, то станет еще страшнее от непонятных ключей и директив.

В файле конфигурации sendmail все параметры сгруппированы по разделам. Разбиение оформляется в виде следующих строк:

```
#####  
#   Local info   #  
#####
```

Столь «красивое» оформление означает, что далее следует раздел `Local info`. Подобных секций несколько — в частности, обычно присутствуют такие:

- ☐ `Local info` — локальная информация, основные сведения о сервере и домене;
- ☐ `Options` — настройки работы программы sendmail;
- ☐ `Message precedences` — приоритеты сообщений;
- ☐ `Trusted users` — доверенные пользователи;
- ☐ `Format of headers` — форматы заголовков.

Рассмотреть все настройки в этой книге невозможно — в моей системе этот конфигурационный файл занимает 50 Кбайт. Если расписывать каждый параметр, понадобится отдельное издание. Наша цель — эффективность и безопасность, поэтому

рассмотрим только те настройки, которые касаются этих вопросов, и испытаем sendmail в действии.

Для упрощения конфигурирования sendmail в последних версиях этого почтового сервиса используется новый файл — `sendmail.mc` (MC означает Minimal Configuration или минимальная конфигурация), который можно найти в каталоге `/etc/mail`. Пример содержимого этого файла приведен в листинге 8.1.

Листинг 8.1. Фрагмент файла `/etc/mail/sendmail.mc`

```
divert(-1)
dnl This is the sendmail macro config file. If you make changes to this
dnl file, you need the sendmail-cf rpm installed and then have to
dnl generate a new /etc/sendmail.cf by running the following command:
dnl Это файл макроконфигурации sendmail. Если вы делаете изменения в этом
dnl файле, вам нужна инсталляция rpm sendmail-cf, с помощью которой можно
dnl сгенерировать новый /etc/sendmail.cf, запустив команду:
dnl
dnl      m4 /etc/mail/sendmail.mc > /etc/sendmail.cf
dnl
include(`/usr/share/sendmail-cf/m4/cf.m4')
VERSIONID('linux setup for ASPLinux')dnl
OSTYPE('linux')
dnl Uncomment and edit the following line if your mail needs to be
dnl sent out through an external mail server:
dnl Раскомментируйте следующую строку, если вам нужно посылать почту
dnl через внешний почтовый сервер:
dnl define(`SMART_HOST', `smtp.your.provider')
define(`confDEF_USER_ID', ``8:12'')dnl
undefine(`UUCP_RELAY')dnl
undefine(`BITNET_RELAY')dnl
...
...
```

Файл `sendmail.mc` имеет более простой формат по сравнению со старым `sendmail.cf`, благодаря чему уменьшается вероятность допустить ошибку при его конфигурировании. После внесения изменений его необходимо скомпилировать (перевести файл `sendmail.mc` в формат `sendmail.cf`) специальной командой, в результате чего получается файл `/etc/sendmail.cf`.

Мы в основном будем рассматривать параметры, которые нужно устанавливать в файле `sendmail.cf`, а если описывается параметр файла `sendmail.mc`, я буду явно указывать на это.

Иногда при неправильных настройках загрузка Linux может зависать при старте сервиса sendmail. Это происходит из-за того, что ваш почтовый сервер не может определить имя компьютера. Откройте файл `/etc/hosts`. В нем чаще всего будет в наличии только одна строка:

```
127.0.0.1      localhost.localdomain      localhost
```

Подробно с этим файлом мы познакомимся в *главе 11*, когда станем рассматривать DNS. Сейчас нам надо знать, что эта строка описывает IP-адрес 127.0.0.1, который соответствует имени localhost. В любой системе такие адрес и имя указывают на вашу локальную машину. Когда в сетевых программах вы указываете localhost, то это имя преобразуется в IP-адрес 127.0.0.1.

Настройка DNS может быть очень важна для правильной отправки почты. Допустим, что у меня есть домен **www.flenov.ru**. Этот домен может загружать веб-сайт, а может отправлять почту, если послать что-либо на почтовый ящик **info@flenov.ru** (в реальности такой ящик не существует). В обоих случаях домен один и тот же, но веб-сервер и почтовый сервер могут находиться на разных серверах с разными IP. Настройка DNS позволяет указать, какой IP-адрес у веб-сервера и какой адрес у почтового.

Для почтового сервера нужно будет в настройки DNS добавить следующие строки:

```
IN MX 10 mx.flenov.ru.  
mx.flenov.ru. IN AAAA ip_адрес_в_формате_v6  
mx.flenov.ru. IN A ip_адрес_в_формате_v4
```

Домен Flenov.ru поменяйте здесь на свой, и точка после него не случайна.

Ну и домен этот должен быть прописан в host-файле:

```
127.0.0.1      flenov.ru
```

Пока мы с DNS еще не знакомы, поэтому в sendmail можно для тестов использовать имя компьютера, которое вы указали во время установки ОС Linux. Выполните команду `hostname`, чтобы это имя узнать. В моем случае это FlenovM. Но на деле все сетевые обращения происходят не по имени, а по IP-адресу, и если sendmail не может определить адрес по имени FlenovM, происходит зависание. Чтобы исправить эту ситуацию, добавьте в файл `/etc/hosts` строку вида:

```
192.168.77.1      FlenovM      FlenovM
```

Не забудьте только заменить приведенный здесь параметр FlenovM на имя вашего компьютера и, разумеется, указать свой IP-адрес. После этого программу sendmail можно помещать в автозапуск, и она будет работать великолепно даже с настройками по умолчанию.

Для каждого пользователя системы автоматически создается почтовый ящик, и если сервис sendmail запущен, то им уже можно пользоваться. Все почтовые ящики хранятся в каталоге `/var/spool/mail/`. Их имена соответствуют именам пользователей. Так, для учетной записи root ящик будет расположен в файле `/var/spool/mail/root`.

Для работы с почтой нам нужен почтовый клиент, который будет отправлять письма серверу и принимать от него новые сообщения. Таких программ существует множество, и в некоторых дистрибутивах Linux предлагается аж семь различных клиентов. Какой выберете вы, зависит только от личных предпочтений.

В общем-то можно обойтись и без клиента, а соединение с сервером осуществлять напрямую через сервис Telnet, благо команды SMTP достаточно просты и ими лег-

Из первой строки видно, что в очереди находится одно сообщение. Вторая строка включает дату отправки и адрес отправителя: `flenov@flenovm.ru`. В последней строке отображается получатель сообщения: `root@flenovm.ru`;

- ❑ `mailstats` — отобразить статистику сообщений и количества байтов;
- ❑ `sendmail` — запустить сервер `sendmail`. Используя эту команду с различными ключами, можно увидеть достаточно много полезной информации. За более подробной справкой следует обратиться к документации: `man sendmail`.

8.4. Безопасность `sendmail`

Безопасность сервиса `sendmail` далека от идеала, и в нем регулярно находят ошибки. По этому поводу администраторы и программисты начали слагать анекдоты и превратили сервис в объект для насмешек. Я слышал, что некоторые даже делали ставки на то, будет ли найдена ошибка в этом месяце или нет.

Как я и обещал, здесь мы поговорим о некоторых параметрах, повышающих безопасность `sendmail`.

8.4.1. Баннер-болтун

Проблемы безопасности начинаются еще на этапе подключения. Сервис `sendmail`, как и большинство других служб, выдает строку приветствия, в которой содержится информация об имени и версии программы.

Хакер не должен знать этих данных. Соответственно, необходимо изменить параметр `SmtpGreetingMessage` в файле `/etc/sendmail.cf`. В старых версиях `sendmail` этот параметр был равен:

```
SmtpGreetingMessage=$j Sendmail $v/$Z; $b
```

Самое опасное здесь — ключ `$v/$Z`, который отображает версию и само имя `Sendmail`. Именно поэтому теперь значение этому параметру присваивается следующим образом:

```
SmtpGreetingMessage=$j $b
```

Если ваша система сообщает о себе что-то лишнее, то это значение необходимо убрать. Можно даже поставить сообщение другого сервиса:

```
SmtpGreetingMessage=$j IIS 5.0.1 $b
```

Любая удачная попытка ввести хакера в заблуждение — это выигрыш во времени, что равносильно маленькой победе.

8.4.2. Только отправка почты

Очень часто почтовые сервисы используют только для отправки почты. Например, на веб-серверах `sendmail` может стоять лишь для того, чтобы можно было отослать

письмо прямо из сценариев на Perl или PHP. Если ваш сервер не должен принимать писем, то необходимо запретить режим приема. Для этого откройте файл `/etc/sysconfig/sendmail` и измените его содержимое:

```
DEAMON=yes  
QUEUE="q1h"
```

Вторая строка задает параметры, которые будут передаваться программе `sendmail` при запуске. Чтобы возобновить прием почты, измените значение `q1h` на `bd`.

Если у вас нет файла `/etc/sysconfig/sendmail` (он используется не во всех дистрибутивах), то придется редактировать сценарий `/etc/rc.d/init.d/sendmail`. Найдите в этом файле параметры, которые передаются программе, и измените их на `q1h` прямо в тексте сценария.

8.4.3. Права доступа

Ни один сервис в ОС не должен работать от имени администратора `root`. Если в коде программы будет найдена лазейка, позволяющая запускать команды, то можно считать систему потерянной, потому что директивы будут выполняться с правами `root`. Опытные пользователи компьютеров, наверное, помнят, что несколько лет назад в `sendmail` чуть ли не каждую неделю находили ошибки и большинство из них были критичными.

Сервис должен работать с правами пользователя, которому доступны только необходимые для работы каталоги и файлы. В последних версиях `sendmail` это уже реализовано с помощью параметра `RunAsUser`:

```
O RunAsUser=sendmail
```

По умолчанию эта строка может быть закомментирована знаком `#` в начале строки. Уберите комментарий. Можно также явно добавить описание группы, с правами которой должна происходить работа:

```
O RunAsUser=sendmail:mail
```

Здесь `sendmail` — это имя пользователя, а `mail` — имя группы.

8.4.4. Лишние команды

Почтовый сервер обрабатывает множество команд, но не все из них могут оказаться полезными. Убедитесь, что в вашем конфигурационном файле присутствуют и не закрыты комментарием следующие строки:

```
O PrivacyOptions=authwarnings  
O PrivacyOptions=noexpn  
O PrivacyOptions=novrfy
```

Можно также в одной команде указать все параметры через запятую:

```
O PrivacyOptions=authwarnings,noexpn,novrfy
```

Наиболее опасной для сервера может оказаться опция `VRFY`, которая позволяет проверить существование почтового ящика. Именно ее запрещает третья строка (третий параметр) в приведенном примере.

Вторая строка устанавливает параметр `noexpn`, запрещающий команду `EXPN`, которая позволяет по почтовому псевдониmu определить адрес E-mail или даже имя пользователя. С помощью этой директивы хакеры могут собирать списки для рассылки спама. Не стоит давать в руки злоумышленника такую информацию.

8.4.5. Выполнение внешних команд

В почтовом сервисе есть одна серьезная проблема — ему необходимо выполнять системные команды, а это всегда опасно. Если хакер сможет запустить такую команду без ведома администратора и с повышенными правами, то это грозит большими неприятностями. Именно поэтому мы понижали права, с которыми работает сервис, но этого недостаточно.

Чтобы запретить выполнение системных команд, необходимо заставить `sendmail` работать через безопасный интерпретатор команд. Для этого специально был разработан интерпретатор `smrsh`. Чтобы почтовый сервис использовал именно его, проще всего добавить в файл `sendmail.mc` следующую строку:

```
FEATURE('smrsh', '/bin/smrsh')
```

Здесь в скобках указано два параметра: имя командного интерпретатора и каталог, в котором он располагается. Убедитесь, что в вашей системе к нему ведет именно такой путь, или измените параметр.

По умолчанию интерпретатор `smrsh` выполняет команды из каталога `/usr/adm/sm.bin`. Программы из других каталогов запускать невозможно. Если в каталоге `/usr/adm/sm.bin` находятся только безопасные программы, то ваша система наименее подвержена уязвимости.

8.4.6. Доверенные пользователи

В сервисе `sendmail` можно создать список пользователей, которым вы доверяете и разрешаете отправлять им сообщения без каких-либо предупреждений. Этот перечень находится в файле `/etc/mail/trusted-users`. Я не рекомендую вам здесь указывать реальных пользователей.

Но файл все же может быть полезен. В него можно добавить пользователя `apache`, чтобы проще было рассылать письма из веб-сценариев.

8.4.7. Отказ от обслуживания

Почтовые серверы довольно часто подвергаются атакам типа DoS, потому что они должны принимать соединения для обслуживаемых почтовых ящиков от любых пользователей. Соответственно, подключения на порты 25 и 110 чаще всего общедоступны.

Для защиты сервера от DoS-атак со стороны хакера нам помогут следующие параметры сервиса sendmail:

- `MaxDeamonChildren` — ограничение количества одновременно запущенных процессов. С помощью этого параметра мы можем защитить ресурсы сервера (процессор) от излишней перегрузки. По умолчанию установлено значение 12, но для мощного компьютера его можно повысить, чтобы эффективнее использовать процессор, а для слабого — уменьшить;
- `ConnectionRateThrottle` — максимальное количество открываемых соединений в секунду. По умолчанию этот параметр равен 3, и повышать его без особой необходимости не стоит, разве что вы уверены в производительности сервера.

8.5. Почтовая бомбардировка

С почтовой бомбардировкой я встретился первый раз более 10 лет назад. Однажды я в чате оставил свой E-mail (до этого я никогда не «светил» своим адресом), и, как назло, в этот момент там сидел начинающий хакер, который просто шутки ради забросал меня почтовыми бомбами.

Что такое почтовая бомба? Это простое письмо с бесполезным содержимым любого размера. Хакеры забрасывают свою жертву такими посланиями, чтобы переполнить его ящик, лишив возможности принимать другие сообщения. Это классическая атака DoS, только в отношении почтового ящика.

На первый взгляд необходимо просто увеличить размер ящика или убрать лимит вообще. Это неверное решение, поэтому забудьте про него. Если ящик не будет иметь предела, то хакер сможет произвести атаку DoS на весь сервер.

Почтовые сообщения — единственный способ закатать информацию на сервер. Когда злоумышленник отправляет сообщение E-mail, оно сохраняется на сервере, пока не будет скачано пользователем. Слишком большое количество информации займет все пространство на жестком диске, и сервер больше не сможет принимать сообщения ни на один из почтовых ящиков.

Самая худшая ситуация при почтовой бомбардировке может возникнуть, когда почтовые ящики располагаются в каталоге по умолчанию (раздел `/var`). Если будет переполнен этот раздел, то сервер не сможет больше записывать в него информацию. А в разделе `/var` хранятся еще и журналы безопасности. Если они не смогут пополняться, то сервер окажется полностью недоступным.

Ограничение на используемое под хранение писем пространство необходимо. Лучше потерять контроль над одним почтовым ящиком, чем над всем почтовым сервером.

От почтовой бомбардировки защититься нельзя, но можно попытаться усложнить злоумышленнику задачу. Для этого нам понадобятся параметры, которые мы рассматривали в *разд. 8.4.7*. Кроме того, желательно ограничить максимальный размер сообщения до приемлемых пределов с помощью параметра `MaxMessageSize`. Если это сделать, злоумышленнику понадобится направлять на сервер множество ма-

леньких писем вместо нескольких больших. Однако, к сожалению, большие письма иногда отправляют не только хакеры, и этим вы усложните жизнь и реальным корреспондентам.

8.6. Спам

Проблема XXI информационного века — рассылка нежелательной корреспонденции. Это действительно болезнь, с которой надо бороться, а существующие методы пока не приносят необходимого результата, и спам отнимает большую часть почтового трафика.

Один из способов борьбы с нежелательной корреспонденцией — запрет серверов, с которых приходит спам. Но те, кто занимаются такими рассылками, находят все новые пути для обхода барьеров, в том числе используют общедоступные или взломанные в Интернете серверы.

Если хакер задействует ваш сервер для рассылки спама, то это грозит следующим:

- ☐ лишние расходы на трафик, если вы оплачиваете каждый гигабайт информации;
- ☐ дополнительная нагрузка на ресурсы. Рассылки в основном массовые и отнимают много процессорного времени, тем самым загружая канал связи.

Но помимо этого, если ваш сервер пару раз разошлет спам, он может попасть в черный список, и тогда вся ваша корреспонденция будет фильтроваться и не дойдет до адресата.

8.6.1. Блокировка приема спама

Прием нежелательной корреспонденции приводит к следующим негативным последствиям:

- ☐ излишние расходы на трафик, которые постоянно увеличиваются;
- ☐ отвлечение внимания сотрудников вашего предприятия или пользователей сети, пользующихся услугами почтового сервера;
- ☐ спам-сообщения нередко занимают слишком много места, и для их хранения на сервере требуется дополнительное дисковое пространство.

Причин борьбы со спамом намного больше, но я надеюсь, что здесь описанных уже достаточно, чтобы вы начали предпринимать какие-то меры.

Фильтрация серверов

В sendmail есть возможность фильтровать серверы, с которых приходит нежелательная почта. Для этого лучше всего в файле sendmail.mc добавить строку с запретом. Проблема в том, что для различных версий sendmail эта строка выглядит по-разному:

Версия 8.10:

```
FEATURE('dnsbl', 'spam.com', ' 550 Mail not accepted from this domain')dn1
```

Версии 8.11 и более поздние:

```
HACK('check_dnsbl', 'spam.com', '', 'general', 'reason')dnl
```

В обеих строках параметр `spam.com` нужно заменить на адрес сервера DNSBL — сервера, на котором хранятся списки IP-адресов (так называемые спам-листы), с которых рассылается спам. Список таких серверов можно посмотреть по адресу: <http://spamlinks.net/filter-dnsbl-lists.htm>. При наличии адреса в списке соединение с ним блокируется. Этот метод не очень эффективен, потому что спам-листы, разумеется, содержат адреса лишь некоторых, уже сильно наследивших, спамеров. К тому же, напротив, были случаи, когда ошибочно блокировались безобидные серверы.

Однажды мой сервер, с которого происходит продажа программного обеспечения, был заблокирован в одном из спам-листов. Это были времена, когда в черные списки попадали и по делу, и просто так. Когда я рассылал своим пользователям их регистрационные ключи, то 10% сообщений возвращалось. В результате некоторые пользователи не могли работать с нашими программами. Это продолжалось в течение месяца, пока составители спам-листа не убедились, что он перестал быть эффективным, и стали искать другие методы.

Фильтрация сообщений

Более эффективный метод — блокировка сообщений по их содержимому. Специализированная программа анализирует всю информацию, которая проходит через сервер, и ищет характерные признаки спам-рассылки. Если определяется, что письмо содержит спам, то оно удаляется.

Этот способ более эффективен, но по тексту очень сложно определить, является ли письмо рекламной рассылкой. Хакеры постоянно ищут новые пути обхода таких блокировок, поэтому процент фильтрации невысок. Вы можете настроить программу так, что она будет уничтожать все сообщения, в которых есть слова «купи», «продаю» и тому подобные, характерные для спама, но тогда могут быть удалены и необходимые вам письма.

Для эффективной борьбы со спамом в Linux есть утилита SpamAssassin. Она по умолчанию не устанавливается, поэтому, чтобы установить ее в Ubuntu, выполняем команду:

```
apt install spamassassin
```

А в CentOS:

```
yum install spamassassin
```

Из Интернета будет скачано около 30 мегабайт установочных файлов, а сама установка потребует до 130 Мбайт на диске.

По завершении установки открываем для редактирования конфигурационный файл `/etc/mail/spamassassin/local.cf` и убираем в нем комментарий со следующей строки:

```
rewrite_header Subject ***SPAM***
```

А также добавляем еще два параметра, если они отсутствуют:

```
required_hits 8.0
report_safe 0
```

Если у вас на сервере электронная почта отправляется одному или двум пользователям, то первый параметр можно понизить до 5. Это порог, после которого письмо будет восприниматься как спам.

Теперь нужно создать пользователя и дать ему права на запись в журнал:

```
useradd -s /bin/false -d /var/log/spamassassin spamd
chown spamd:spamd /var/log/spamassassin
```

При создании пользователя `spamd` я с помощью параметра `-s` задал командную оболочку `/bin/false`, чтобы этот пользователь не смог войти в систему. Нам же он нужен лишь для того, чтобы дать права доступа к журналам.

Последняя строка предоставляет пользователю `spamd` доступ к папке `/var/log/spamassassin`.

Теперь открываем конфигурационный файл `/etc/default/spamassassin`, в котором нужно установить следующие параметры:

```
ENABLED=1
CRON=1
SAHOME="/var/log/spamassassin/"
OPTIONS="--create-prefs --max-children 2 --username spamd \-H ${SAHOME} -s
${SAHOME}spamd.log"
```

На этом конфигурация закончена, можно запустить сервис:

```
service spamassassin start
```

Сервис `spamassassin` готов фильтровать вашу почту, но это всего лишь фильтр. Он не умеет читать почту и не умеет доставлять ее. Очень часто для получения почты используется сервис `postfix`. Новое название? Мы уже рассмотрели сервис `sendmail`, который отправляет почту, и сервис `spamassassin`, который ее фильтрует. А сервис `postfix` почту принимает (более подробно сервис `postfix` описан в *разд. 8.7*).

Настройки сервиса `postfix` находятся в файле `/etc/postfix/master.cf`. Откройте его в текстовом редакторе и найдите строку:

```
smtp      inet  n       -       -       -       smtpd
```

Добавьте в конец этой строки следующий текст:

```
-o content_filter=spamassassin
```

А также добавьте еще одну строку:

```
spamassassin unix  -       n       n       -       -       pipe user=spamd
argv=/usr/bin/spamc -f -e /usr/sbin/sendmail -oi -f ${sender} ${recipient}
```

Чтобы изменения вступили в силу, `postfix` нужно перезапустить:

```
service postfix restart
```

Я пользуюсь сервисом spamassassin уже свыше трех лет, и только два раза вытаскивал из папки спама важные письма. Также очень редко что-либо ненужное просачивается в папку входящих моего рабочего ящика. В целом я очень доволен этим фильтром.

8.6.2. Блокировка пересылки спама

При конфигурации почтового сервиса вы должны сделать так, чтобы злоумышленники не смогли посылать свой спам через ваш сервер. Чтобы такая массовая рассылка перестала быть эффективной, вам необходимо произвести несколько настроек:

- по умолчанию протокол SMTP не требует авторизации, поэтому любой пользователь может подключиться к серверу и отправить письмо кому угодно. Чтобы избежать этого, можно выполнить одно из следующих действий:
 - запретить с помощью сетевого экрана подключение к порту SMTP пользователей, которые находятся вне вашей сети. Такую защиту чаще всего используют провайдеры и администраторы частных или корпоративных сетей. С реализацией этого запрета у вас не должно возникнуть проблем, потому что мы уже не раз ее рассматривали;
 - разрешать отправку почты только в течение определенного времени (например, 10 минут) после проверки почты по протоколу POP3. В момент проверки почты сервер производит авторизацию клиента, и по этим данным в сетевом экране может создаваться временная разрешающая запись для доступа к SMTP (можно применить и другой какой-либо способ). Теперь в течение 10 минут с этого IP-адреса можно будет отправлять почту;
 - использовать авторизацию SMTP. Изначально в стандарте на протокол отправки сообщений нет ничего об идентификации пользователя, поэтому не все серверы ее поддерживают. Но в sendmail и других мощных пакетах есть расширение, которое позволяет реализовать авторизацию и сделать ее обязательной;
- запретить отправку слишком большого количества писем с одного и того же IP-адреса. Вполне нормальным числом является 20. Пользователь не должен иметь права посылать более 20 писем за 10 минут;
- запретить отправку писем большому количеству получателей. Письмо может содержать в поле **Копия** список пользователей, которым направлено письмо. Желательно установить ограничение на размер этого списка.

Существуют и другие методы, но даже этих будет достаточно.

В последних версиях sendmail по умолчанию разрешается пересылка почты только с тех компьютеров, которые прописаны в файле /etc/mail/access. В листинге 8.2 приведено содержимое этого файла.

Листинг 8.2. Файл /etc/mail/access

```
# Check the /usr/share/doc/sendmail/README.cf file for a description
# of the format of this file. (search for access_db in that file)
# The /usr/share/doc/sendmail/README.cf is part of the sendmail-doc
# package.
#
# Смотрите файл /usr/share/doc/sendmail/README.cf для получения
# информации по формату файла (ищите слово access_db в этом файле)
#
# by default we allow relaying from localhost...
# По умолчанию мы разрешаем рассылку только с локального хоста
localhost.localdomain      RELAY
localhost                  RELAY
127.0.0.1                  RELAY
```

В этом файле можно оставить разрешение отсылать почту только с компьютеров внутренней сети или с самого сервера. Для этого файл должен содержать следующие записи:

```
localhost      RELAY
your_domain.com RELAY
```

Все остальные не смогут выполнять рассылку писем. Такой метод хорош только в том случае, когда серверы и компьютеры защищены. Если хакер проник в сеть, то он сможет разослать любой спам от имени пользователя сети. От этого никуда не деться. Если есть хоть какое-то разрешение, то им можно воспользоваться, и ваша задача сделать так, чтобы это было, по крайней мере, сложно.

Запрет рассылки не всегда может быть реализован, поэтому приходится использовать другие методы — например, заставить пользователей проверять почту по протоколу POP до отправки писем. Для осуществления этого метода можно воспользоваться сервисом pop-before-smtp (popsmtplib.sourceforge.net), который проверяет журнал сообщений /var/log/maillog, и отправка почты разрешается, только если в нем найдена удачная авторизация за определенный период.

Единственный, но существенный недостаток этого способа заключается в том, что если на пути следования письма стоит анонимный прокси-сервер или маскирующий сетевой экран, то пакеты будут приходить с другим IP-адресом. Это значит, что все пользователи, которые подключены через тот же прокси или брандмауэр, также автоматически считаются авторизованными. Таким образом, поиск по IP-адресу записей в журнале не может дать полной защиты.

Наиболее предпочтительным является метод, при котором используется SMTP-авторизация (SMTP AUTH), которая описана в RFC 2554. В сервисе sendmail поддержка этого расширения существует еще с версии 8.10.

Если вы решили использовать SMTP AUTH, то убедитесь, что почтовые клиенты пользователей имеют возможность авторизации на сервере и при этом настроены на ее использование.

8.7. Сервер Postfix

В моей любимой Mandriva используется более простой в настройке почтовый сервер, который называется Postfix. Его можно также встретить в Debian и SUSE. За конфигурацию этого сервера отвечает файл `/etc/postfix/main.cf`. Здесь параметров не так много, потому что вся конфигурация сервера разбита на несколько файлов, и `main.cf` является главным, но не единственным. Например, более полная версия этого файла — `main.cf.dist`.

Мы рассмотрим здесь лишь базовые параметры конфигурации сервера Postfix. Знание английского и хорошие комментарии в файле конфигурации помогут вам разобраться со многими значениями, если вдруг понадобится что-то из того, что мы не рассмотрели. Я же постарался выбрать наиболее важные параметры, которые покроют максимальное количество ваших потребностей.

Первый параметр, который может пригодиться, — `myorigin`. Как и большинство базовых параметров, его можно найти в файле `main.cf.dist`. Там вы увидите аж две строки следующего вида:

```
#myorigin = $myhostname
#myorigin = $mydomain
```

Обе строки закомментированы — они лишь предлагают вам выбрать тот вариант, который лучше подходит. Чтобы одну из строк сделать активной, просто убираем символ комментария `#` в начале строки. После символа равенства идет значение, но что же это за интересные значения, начинающиеся с символа доллара? Это переменные. Все, что начинается с доллара, — это переменные, и вы можете создавать собственные переменные и определять им значения. Например, в следующей строке кода я создал переменную `$siteofflenov` со значением `www.flenov.info`:

```
$siteofflenov = www.flenov.info
```

Теперь посмотрим на переменные, которые мы недавно увидели: `$myhostname` и `$mydomain`. По имени уже можно понять, что первая равна имени хоста, а вторая — домену. Так что же означает параметр `myorigin`? Он определяет, как сервер будет представляться при работе с другими системами.

Следующий параметр — `inet_interfaces`. С его помощью можно указать интерфейсы, с которых нужно ожидать почту. По умолчанию сервер ожидает письма со всех активных интерфейсов, и параметр равен значению `all`. Но иногда необходимо запретить определенные интерфейсы.

Параметр `mydestination` определяет домены, для которых сервер принимает почту.

8.7.1. Псевдонимы

Для настройки псевдонимов служит файл `aliases`. В этом файле находятся записи вида:

```
admin: root
```

Слева от двоеточия стоит псевдоним (алиас), а справа можно увидеть имя пользователя. В нашем случае имя `admin` является алиасом для `root` — другими словами, все письма, которые будут приходить на имя `admin`, станут автоматически пересылаться на ящик `root`.

Нужно иметь в виду, что файл `aliases` является конфигурационным и текстовым, а сервер использует его бинарную версию `aliases.db`. Чтобы создать этот db-файл, необходимо выполнить команду:

```
postalias aliases
```

Но и это еще не все. Изменения вступают в силу не сразу, а только по прошествии какого-то времени. Если вы торопитесь и хотите получить результат немедленно, можно перезапустить сервер. Это помогает всегда. Но неужели после каждого изменения нужно перезапускать программу? Конечно же, нет, это не самое лучшее решение, потому что есть способ проще — выполните следующую команду:

```
postfix reload
```

8.7.2. Ретрансляция

Сервер Postfix может передавать письма, только если:

- ☐ отправитель принадлежит одной из сетей, указанных в переменной `$mynetworks`. По умолчанию эта переменная равна всем сетям, которые подключены к данному компьютеру, например:

```
mynetworks = 168.100.189.0/28, 127.0.0.0/8
```
- ☐ домен отправителя или домен получателя сообщения присутствует в переменной `$relay_domains`.

Такая защита необходима, чтобы спамеры не использовали ваш почтовый сервер в качестве средства распространения спама. За спам в наше время сильно наказывают. Нет, в тюрьму сажают редко, но в «игнор» помещают без разговора. А если сервер попадет в список игнорируемых, то вы не сможете отправлять никаких сообщений.



ГЛАВА 9

Шлюз в Интернет

Мы уже знаем, как установить и настроить сервер Linux, сделать его соединение безопасным и подключить основные сервисы. Но такие службы, как веб-сервер и электронная почта, используются не только в локальной сети. Их максимальные преимущества проявляются при интеграции с Всемирной сетью.

Когда мы строим дом, то уже при возведении стен заботимся о безопасности, устанавливая окна и двери, думаем об их надежности, а закончив отделку дома, ставим сигнализацию. Дом мы уже построили и сделали его защищенным. Именно поэтому безопасность сети рассматривалась первой. Теперь нам необходимо поставить двери, через которые можно будет выходить из дома на улицу (в Интернет). После завершения всех настроек мы повесим на нашу обитель сигнализацию — системы мониторинга и выявления атак, которые будут рассматриваться в *главе 12*.

В качестве дверей во Всемирную сеть будут выступать шлюз и прокси-сервер (Проху). Именно о них пойдет речь в этой главе, и как раз их мы будем подробно рассматривать. Но настройка связи не заканчивается конфигурированием сервера, на клиенте тоже необходимо производить определенные действия, с которыми нам также предстоит познакомиться.

9.1. Работа прокси-сервера

Изначально прокси-серверы (Проху) создавались для решения узкого круга задач, а именно — для кэширования данных, получаемых из Интернета. Например, сотрудники вашей фирмы работают на 100 компьютерах, которые подключаются к Интернету через один физический канал связи. Не секрет, что большая часть пользователей загружает по нескольку раз в день одни и те же страницы, и каждый раз эта загрузка «давит» на канал связи и трафик.

Сделаем простейший расчет. Ежедневно мы обращаемся к поисковым системам — например, **www.yahoo.com** или **www.google.com**. Теперь подсчитайте, сколько происходит загрузок сайта **www.yahoo.com** со 100 компьютеров, если учесть, что в среднем человек обращается к поисковикам не менее 10 раз в день для поиска

разной информации и уточнения запросов. Получится число более 1000. При этом если сами страницы меняются в зависимости от запроса, то, скажем, картинки остаются одними и теми же. Таким образом, трафик расходуется напрасно, каждый раз передавая одну и ту же картинку.

Для экономии интернет-трафика и были придуманы прокси-серверы. Со временем их возможности стали наращиваться, и в настоящее время можно выделить следующие преимущества от использования подобных программ:

- ☐ кэширование документов, получаемых по сети;
- ☐ кэширование результатов DNS-запросов;
- ☐ организация шлюза доступа в сеть;
- ☐ управление доступом в Интернет;
- ☐ анонимный доступ в сеть через сокрытие адреса;
- ☐ экономия IP-адресов.

В этой главе мы поговорим о самом популярном в Linux прокси-сервере squid, рассмотрим его возможности, оценим безопасность и познакомимся с конфигурированием.

Чтобы сэкономить трафик и одновременно увеличить скорость загрузки на сервере, через который осуществляется выход в Интернет, устанавливается специализированная программа (прокси-сервер), которая кэширует весь трафик (рис. 9.1). Когда первый пользователь загружает страницу **www.yahoo.com**, то все ее содержимое сохраняется в кэше прокси-сервера. При следующем обращении к тому же веб-узлу все картинки скачиваются уже не из Интернета, а с прокси-сервера, а текстовая часть (в зависимости от содержимого страницы и происшедших изменений) может быть загружена с сервера.

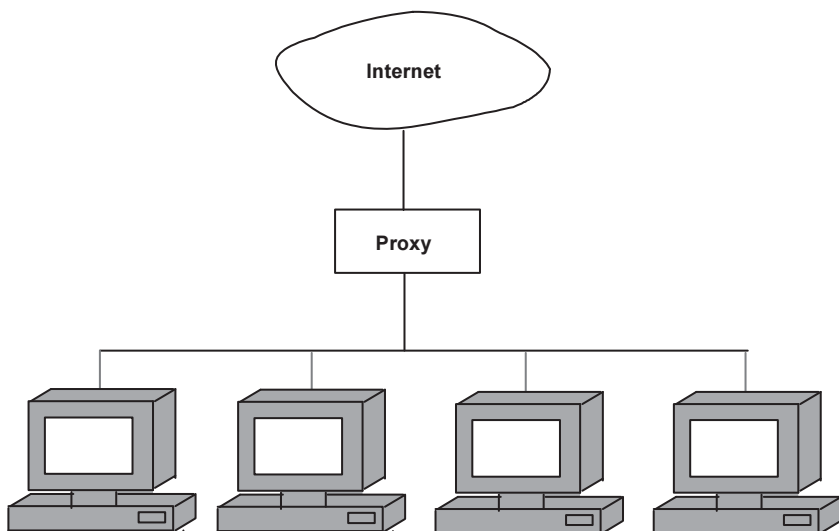


Рис. 9.1. Организация работы с Интернетом через прокси-сервер

Как правило, на сайтах именно графический материал занимает наибольший объем. Если текстовая часть страниц обычно не превышает 15 Кбайт, то общий размер графики достигает 100 Кбайт и более. Загружая эту информацию с локального прокси-сервера, вы экономите трафик и время.

Скорость загрузки увеличивается за счет того, что прокси-сервер находится в вашей локальной сети, и связь с ним чаще всего быстрая. Реальная скорость соответствует вашему оборудованию, пропускная способность которого в настоящее время даже в самых дешевых вариантах достигает 100 Мбит/с. По этому каналу вы забираете большую часть информации: всю графику, флеш-анимацию и неизмененную текстовую часть. Связь с Интернетом намного медленнее, и в малых офисах в среднем составляет от 54 Мбит/с. Такой низкой скоростью может быть и при использовании дешевого оборудования Wi-Fi. И чем дальше вы от передатчика, тем хуже прием и скорость. Но через этот канал вы забираете только измененные текстовые данные (чаще всего содержимое HTML-файлов).

Помимо кэширования содержимого страниц, прокси-сервер может сохранять результаты DNS-запросов. Это также способно повысить производительность. Пользователю удобнее вводить символьные адреса, а компьютер обменивается данными, используя IP-адреса. Соответственно, прежде чем начнется загрузка, программа должна выполнить такую подмену, что занимает какое-то время и создает задержку перед началом обмена данными. Если до вас уже кто-нибудь обращался к сайту по символьному имени, то задержка на работу с сервером DNS будет меньше, потому что прокси-сервер возьмет адрес из своего кэша, не обращаясь к удаленному серверу. Более подробно о DNS мы поговорим в *главе 11*.

С развитием Интернета и потребностей пользователей стали расти и возможности прокси-серверов. Прокси-сервер может играть роль шлюза и без дополнительных программ или оборудования обеспечить доступ в Интернет. Помимо этого, он становится щитом в сети от вторжения извне. Например, если все пользователи подключены к Интернету через прокси-сервер, а он прячет реальный IP-адрес пакетов и отправляет их в сеть от своего имени, то хакеры увидят только IP-адрес прокси-сервера и будут ломать его, а компьютеры реальных пользователей останутся незатронутыми. Таким образом, при организации защиты от внешнего вторжения можно больше внимания уделять охране прокси-сервера, чем клиентских компьютеров.

Возможность сокрытия IP-адреса дает еще одно преимущество — экономия адресов. Интернет-адрес должен быть только у прокси-сервера, потому что именно он обменивается пакетами с внешним миром. Все остальные компьютеры в вашей локальной сети могут иметь немаршрутизируемые адреса, которые зарезервированы для частных сетей (диапазон 192.168.x.x или 10.x.x.x).

Прокси-серверы бывают прозрачными и анонимными. *Прозрачные* просто пересылают пакеты пользователя (без изменения адреса отправителя) дальше на веб-сервер. Прокси-сервер, который скрывает IP-адрес, называется *анонимным*. Такой сервер общается с внешним миром от своего имени. Этим очень часто пользуются злоумышленники. Например, если хакер хочет вскрыть сервер и замести следы, то он производит все свои действия через анонимный прокси-сервер, чтобы администратор не смог узнать, кто именно производил взлом.

В настоящее время в Интернете работает множество анонимных прокси-серверов, но не все из них реально прячут адрес. В отдельных случаях источник остается доступным для удаленной системы, а некоторые серверы сохраняют всю активность в журналах, и их могут просмотреть правоохранительные органы. Таким образом, злоумышленник не может быть уверенным, что используемый им сервер действительно анонимен.

У публичных прокси-серверов Интернета есть еще один недостаток — нельзя гарантировать, что на нем нет никакой «заразы». Например, некоторое время назад у нас на работе администратор запретил доступ к сайтам **vkontakte.ru** и **odnoklasniki.ru**. Я думаю, многие из вас войдут в мое положение и поймут горе пользователей. Этот запрет легко обходится с помощью прокси, но проблема в том, что первый же публичный сервер, который я нашел, одарил некоторых пользователей сети целой кучей вирусов и программ-шпионов.

Поскольку не все компьютеры в сети должны иметь право работать с Интернетом, то на уровне прокси-сервера можно производить аутентификацию пользователя.

В некоторых версиях прокси есть очень удобная возможность — обмен информацией между серверами. Например, в здании в одной большой сети находятся несколько офисов, каждый из них платит за Интернет отдельно и общается с внешним миром через свой прокси-сервер. Можно объединить несколько прокси-серверов, и если на одном нет в кэше нужного сайта, то он попытается взять информацию с соседнего сервера.

Чаще всего для реализации такой возможности применяется протокол ICP (Internet Cache Protocol, протокол интернет-кэширования). Если ваш сервер не нашел нужного документа, то он направляет ICP-запрос другим серверам. Если какой-либо прокси-сервер ответит положительно, то информация будет взята у него.

При использовании протокола ICP (или иного способа поиска данных в других прокси-серверах) выигрыш в скорости загрузки не столь заметен при обращении к документам маленького размера, потому что возрастают расходы времени на ICP и поиск информации в кэше. При большой нагрузке на серверы и большой базе кэша поиск может оказаться слишком долгим, и преимущество в скорости исчезает. Единственное, чем полезно такое решение, — экономия трафика, которая может сберечь деньги тем, кто оплачивает каждый получаемый мегабайт.

Мы рассмотрели основные возможности прокси-серверов, но это не значит, что все они есть в любом сервере. Все зависит от разработчика, и некоторые реализуют только одну задачу.

По умолчанию все программы отправляют свой трафик напрямую, и чтобы направлять его через прокси-сервер, это нужно указать в ее настройках. Все существующие программы описывать в книге смысла нет, потому что постоянно выходят новые версии, и что-то меняется с точки зрения интерфейса. Обычно в программе имеется окно свойств, и параметр **Proxy Server** (Прокси-сервер) следует искать там где-нибудь в разделе **Connection** (Соединение).

Под каждую задачу, поддерживающую определенный протокол, как правило, выделяется отдельный порт. Для протокола HTTP, применяемого для загрузки веб-

страниц, чаще всего используются порты 3128 или 8080, но эти значения зависят от сервера и могут быть изменены. Перед использованием определенной программы прокси-сервера убедитесь, что она обладает необходимыми вам возможностями и обеспечивает поддержку всех нужных протоколов. Неподдерживаемые протоколы придется направлять не через сервер, а напрямую, через шлюз.

Для повышения безопасности вашей сети необходимо с помощью сетевого экрана запретить подключения извне на используемые сервисом squid порты. Например, для работы с протоколом HTTP по умолчанию он работает через порт 3128. И если к этому порту будет разрешено подключаться только из локальной сети, то хакер не сможет воспользоваться этим прокси-сервером в своих целях или для получения доступа к компьютерам защищенной им сети.

9.2. Кэширование

Я полагаю, что сейчас не так часто в каких-то сетях устанавливается прокси-сервер для кэширования трафика, приходящего из Интернета. Каналы стали настолько широкими и скоростными, а Интернет настолько огромным и динамично развивающимся, что кэшировать что-либо практически бесполезно. Страницы новостных лент меняются постоянно, и если в какой-то момент закэшировать графику той или иной страницы, то через час на этой странице графика окажется уже другой, и из кэша будет доставлено пользователю всего несколько статичных файлов. Стоит эта экономия трафика затрат на сервер и его сопровождение? Трудно сказать...

Мне кажется, что сейчас прокси-сервер чаще используется владельцами сайтов с особенно высокой нагрузкой. Этот сервис очень быстрый и неплохо справляется в качестве первой линии обороны против наплыва большого трафика. По крайней мере, я использую squid именно так, и давайте посмотрим, как это работает.

Этот сервис устанавливается на серверы, которые принимают запросы от пользователя. Если запрашиваемый файл есть в кэше, то он тут же возвращается пользователю, экономя ресурсы остальных серверов. Если файла нет, то он запрашивается уже у одного из веб-серверов. А их у больших сайтов может быть несколько.

9.3. Прокси-сервер squid

Как я уже отмечал, самым распространенным прокси-сервером является squid. Этот сервер имеет весьма длинную историю, и за время его существования в нем реализовано много возможностей.

Основной конфигурационный файл для squid — `/etc/squid/squid.conf` (в некоторых системах место его расположения `/etc/squid.conf`). Файл очень велик, и приводить его полностью нет смысла, т. к. значительную его часть занимают подробные комментарии по использованию директив.

Рассмотрим основные команды, которые доступны нам для управления прокси-сервером. Как обычно, все параметры, влияющие на производительность и безо-

пасность, мы изучим подробно. Остальные настройки будут приведены более поверхностно — с ними можно поближе познакомиться, прочитав комментарии из конфигурационного файла.

9.3.1. Директивы настройки HTTP

При подключении к Интернету пользователи первым делом загружают веб-страницы. Если используется squid, то необходимо правильно настроить протокол HTTP. Для решения этой задачи в файле `squid.conf` есть следующие директивы:

- `http_port n` — номер порта, через который будет происходить подключение. Порты, на которых сервер будет ожидать подключения клиентов, — это первое, что необходимо настроить. Такие директивы имеют формат `xxxx_port`. Для порта HTTP запись будет выглядеть таким образом:

```
http_port 8080
```

При конфигурировании браузера на клиентском компьютере вы должны будете указать IP-адрес сервера, где установлен squid, и прописанный в этой директиве порт;

- `hierarchy_stoplist` — определяет перечень URL-адресов, данные с которых всегда должны получаться с сервера, а не из кэша. Я рекомендую добавить в этот список слова `cgi-bin` и вопросительный знак. Адреса URL, содержащие такой текст, указывают на сценарии, которые могут исполняться на сервере, и их результат желательно не кэшировать.

Рассмотрим пример. Предположим, что вы прочитали веб-страницу **`www.servername.com/cgi-bin/ping.cgi`**, на которой можно через веб-интерфейс выполнить команду `ping`. Допустим, что при первом обращении вы запустили команду `ping` к адресу `18.1.1.1`. Результат будет сохранен в кэше прокси-сервера. В следующий раз вы обращаетесь к сценарию, чтобы выполнить `ping 18.1.1.18`, но браузер вернет первый результат, потому что возьмет его из своего кэша.

Страницы со сценариями могут возвращать разный результат в зависимости от ситуации и параметров, которые выбрал пользователь. Если же такие страницы кэшировать, то вы всегда будете видеть одно и то же. В результате от соединения через прокси-сервер вы получите только неудобства.

Вопросительный знак очень часто используется для передачи параметров, поэтому такие страницы тоже не рекомендуются кэшировать.

Тег `hierarchy_stoplist` запрещает брать страницу из кэша, а следующие две строки задают правило, по которому страницы с URL-адресом, содержащие слова `cgi-bin` или вопросительный знак, вообще не будут кэшироваться:

```
acl QUERY urlpath_regex cgi-bin \?  
no_cache deny QUERY
```

Думаю, вы согласитесь со мной, что незачем кэшировать то, что все равно при новом запросе будет получаться с сервера, и зря расходовать дисковое пространство.

9.3.2. Директивы настройки FTP

Для работы по протоколу FTP тоже есть несколько директив:

- ❑ `ftp_passive` параметр — режим работы. Если в качестве параметра указано значение `on` (устанавливается по умолчанию), то пассивный режим разрешен.

Сервер `squid` позволяет работать с протоколом FTP, но для этого может потребоваться дополнительная настройка. Например, если `squid` находится за сетевым экраном, запрещающим пассивный режим, то лучше изменить значение параметра по умолчанию, установив для этого следующую директиву:

```
ftp_passive off
```

- ❑ `ftp_user` адрес — определяет адрес E-mail, который будет использоваться в качестве пароля при авторизации на анонимном FTP-сервере.

Ни один сервер не может точно сказать, правильно ли вы указали адрес, поэтому проверка может быть отключена. Но некоторые FTP-серверы проверяют корректность написания адреса. По умолчанию `squid` использует в качестве E-mail слово `squid@`:

```
ftp_user squid@
```

По умолчанию в файле `/etc/squid/squid.conf` эта строка закомментирована, но желательно поменять в ней адрес E-mail, например:

```
ftp_user squid@hotmail.com
```

Такой адрес любой FTP-сервер воспримет как корректный, потому что он соответствует всем правилам написания E-mail;

- ❑ `ftp_list_width n` — число `n` задает ширину листинга при просмотре содержимого FTP-сервера. Это значение должно быть достаточным, чтобы увидеть все файлы. Если установить слишком маленькое значение, то имена файлов будут обрезаться.

9.3.3. Настройка кэша

От того, как вы настроите кэш, зависит удобство работы через прокси-сервер, поэтому я постараюсь показать все соответствующие директивы и подробно рассмотреть каждую из них:

- ❑ `cache_dir` тип каталог размер `L1 L2` опции — определяет параметры каталога, в котором будет храниться кэш. Основными для нас являются тип, каталог и размер. В большинстве случаев для типа применяется значение `ufs`, но если вы используете асинхронный ввод/вывод (я не советую, потому что это может вызвать проблемы в работе), то нужно установить `aufs`.

Каталог надо выбрать в самом большом разделе, чтобы информация не разбрасывалась по нескольким дискам. Если же у вас используется один диск с одним разделом, то расположение каталога не имеет особого значения.

Размер каталога по умолчанию равен 100 Мбайт. Этого достаточно для ускорения работы трех пользователей. Если в вашей сети много пользователей и у каждого свои вкусы (любимые сайты), то значение желательно увеличить. Я использую не менее 1 Гбайт кэша. Выделенное пространство быстро исчезает, особенно если серверу разрешено кэшировать большие файлы;

- ❑ `cache_mem n MB` — задает максимальный размер оперативной памяти, необходимый для программы. По умолчанию задано 8 Мбайт. Если настраиваемый компьютер решает только задачи прокси-сервера, то можно указать значение, равное разнице объемов оперативной памяти и памяти, необходимой для ОС. Для ОС в текстовом режиме 64 Мбайт будет более чем достаточно, и, например, если у вас ОЗУ 512 Мбайт, то 448 Мбайт можно отдать прокси-серверу — чем больше у него оперативной памяти, тем быстрее он сможет отвечать на часто повторяемые запросы;
- ❑ `cache_swap_low n` — процент заполнения кэша. Когда размер кэша превышает значение `n`, сервер начинает его чистить, убирая устаревшие объекты, пока размер не станет удовлетворять параметру;
- ❑ `cache_swap_high n` — процент заполнения кэша. Команда аналогична предыдущей, но сервер начинает освобождать кэш более интенсивно. Это необходимо, чтобы не возникла ситуация, когда кэш будет переполнен;
- ❑ `minimum_object_size n KB` — минимальный размер объекта, который попадает в кэш. По умолчанию установлено значение 0 — при нем порог отсутствует;
- ❑ `maximum_object_size n KB` — максимальный размер объекта, который должен кэшироваться. По умолчанию стоит значение 4096 Кбайт, что соответствует 4 Мбайт. Для повышения производительности сервера необходимо понизить это значение, но при этом может увеличиться расход трафика. Если экономия трафика важнее производительности, то значение `n` лучше увеличить;
- ❑ `maximum_object_size_in_memory n KB` — максимальный размер объекта в памяти. По умолчанию установлено значение 8 Кбайт;
- ❑ `ipcache_size n` — размер кэша для хранения IP-адресов. По умолчанию задано значение 1024 Кбайт;
- ❑ `ipcache_low n` и `ipcache_high n` — соответственно минимальный и максимальный проценты заполнения кэша для IP-адресов;
- ❑ `reference_age` параметр — время жизни объекта в кэше. Если объект пролежал дольше, то его можно удалять по старости. Рассмотрим несколько примеров использования директивы:

```
reference_age 1 week
reference_age 3.5 days
reference_age 4 months
reference_age 2.2 hours
```

По умолчанию используется значение в один год:

```
reference_age 1 year
```

- ❑ `quick_abort_min n KB` — минимальный размер объекта, при котором в случае обрыва соединения необходимо после его восстановления закончить скачивание объекта и полностью его сохранить. Это позволяет сократить трафик и увеличить скорость работы в сети. Например, пользователь запустил на скачивание файл для проверки соединения и оборвал связь. Если сервер успел сохранить файл, то при повторной попытке не надо снова скачивать те же данные. Достаточно их взять из кэша. По умолчанию установлено значение 16. Чтобы отключить эту возможность, можно задать значение -1;
- ❑ `quick_abort_max n KB` — максимальный остаток объекта, при котором загрузка будет прервана в случае обрыва соединения. По умолчанию установлено значение 16;
- ❑ `quick_abort_pct n` — параметр аналогичен `quick_abort_min`, но в этом случае указывается процент уже полученной информации;
- ❑ `negative_ttl n minutes` — количество минут, которые нужно кэшировать негативный ответ сервера. Например, пользователь зашел на сервер и получил ошибку, которая может быть временной, поэтому нельзя кэшировать ответ на длительный срок. Значение по умолчанию — 5 минут. Если пользователь обратится по тому же адресу по истечении этого времени, то копия из кэша не будет использоваться, а произойдет новая попытка зайти на сайт;
- ❑ `positive_dns_ttl n hours` — время в часах, в течение которого нужно кэшировать положительный результат DNS-запроса. В этот промежуток времени при повторных обращениях к DNS IP-адрес будет взят из кэша. По умолчанию задано значение 6 часов, в настоящее время его можно увеличить до 24 часов. Несколько лет назад IP-адреса имели тенденцию очень часто меняться, поэтому приходилось ограничивать время жизни запросов. Сейчас большинство сайтов имеют статические адреса, которые изменяются только при смене хостинга, а крупные порталы зарезервировали себе собственные постоянные IP-адреса. Если вы не хотите использовать кэширование IP-адресов, встроенное в squid, то можно установить этот параметр в 0;
- ❑ `negative_dns_ttl n minutes` — промежуток времени в минутах, в пределах которого нужно кэшировать негативный ответ DNS-сервера. Если по имени не найден IP-адрес, то, возможно, проблема с сервером имен, а не с именем. Такие вопросы чаще всего решаются в течение 2–3 минут, поэтому отрицательный ответ не стоит держать в кэше дольше, иначе все это время клиенты не смогут обратиться к сайту. Я делаю этот параметр равным 1 или 0, чтобы пользователи увидели нужный сайт сразу после устранения проблемы;
- ❑ `range_offset_limit n KB` — порядок передачи клиенту неполных данных. Пользователь может запросить не весь файл, а лишь его часть. В этом случае squid не может кэшировать такой файл, не скачав его целиком. Однако если запрошенная часть сильно смещена от начала файла, то это приведет к большой задержке

перед тем, как squid начнет возвращать что-либо клиенту. Для контроля над такими ситуациями и предусмотрен этот параметр — если смещение запрошенной части более параметра *n*, то прокси-сервер запросит у сервера в Интернете лишь необходимую пользователю часть файла и не станет его кэшировать. Если указать значение *-1*, то squid будет всегда загружать весь файл и кэшировать его содержимое. Если же указать значение *0* (по умолчанию), то, напротив, squid будет скачивать только запрошенные части файлов и кэшировать их, лишь если запрошен весь файл.

9.3.4. Журналы

В конфигурационном файле есть несколько параметров, влияющих на работу прокси-сервера с журналом (журналы легко читаются в любом текстовом редакторе):

- ❑ `cache_access_log` файл — журнал, в котором сохраняется вся активность пользователей, а именно: HTTP- и ICP-запросы. По умолчанию этот параметр равен `/var/log/squid/access.log`;
- ❑ `cache_log` файл — файл для хранения основной информации о кэше. По умолчанию это `/var/log/squid/cache.log`;
- ❑ `cache_store_log` файл — журнал операций над объектами в кэше (убраны или помещены на какое-то время). По умолчанию используется файл `/var/log/squid/store.log`, но вы без проблем можете отключить этот журнал, указав в качестве значения `none`, потому что утилита для анализа сохраняемых данных нет, да и вообще пользы в таких данных мало, а расходы на их сохранение присутствуют;
- ❑ `log_mime_hdrs` параметр — если в качестве параметра указано `on`, то в журнале `access` будут сохраняться заголовки MIME;
- ❑ `useragent_log` — журнал, в котором сохраняется поле `User-agent` заголовков HTTP. Смысла в этом поле нет, потому что его легко подделать, и ничего полезного журнал не даст, поэтому по умолчанию он не используется.

В *разд. 12.5* мы будем говорить о журналах различных сервисов Linux, а в *разд. 12.5.4* подробно рассмотрим содержимое основного журнала squid — `/var/log/squid/access.log`.

9.3.5. Разделение кэша

Чтобы ваш сервер мог обмениваться запросами с другими squid-серверами, разделяя таким образом содержимое кэша, вы должны настроить соответствующий протокол. Для этого есть следующие директивы:

- ❑ `icp_port n` — номер порта, который будет использоваться для ICP-протокола. По умолчанию задано значение `3130`. Если указать `0`, то протокол будет заблокирован;
- ❑ `htcp_port n` — номер порта, который будет использоваться для ICP-протокола, работающего поверх TCP/IP. По умолчанию задано значение `4827`. Если указать `0`, то протокол будет заблокирован;

❑ `cache_peer` адрес тип `http_порт` `icp_порт` опции — сервер, с которым можно обмениваться информацией. В качестве адреса указывается имя (или адрес) сервера, с которым предполагается взаимодействие. Параметр `http_порт` определяет порт, на котором настроен HTTP-прокси, и соответствует параметру `http_порт` в файле конфигурации `squid`. Атрибут `icp_порт` определяет порт, на котором настроен протокол ICP, и соответствует параметру `icp_порт` в файле конфигурации `squid` удаленной системы. В качестве типа может указываться одно из следующих значений:

- `parent` — старший в иерархии;
- `sibling` — равнозначный;
- `multicast` — широковещательный.

Последний параметр опции может принимать много различных значений, поэтому мы его рассматривать не будем, чтобы не утяжелять книгу. В комментариях, содержащихся в конфигурационном файле, каждый параметр подробно описан;

❑ `icp_query_timeout n` — время ожидания в миллисекундах. Чаще всего прокси-серверы расположены в локальной сети с высокой скоростью доступа, и ожидание более 2000 мс будет лишним. Иначе, если ответ не будет получен и придется обращаться в Интернет, пользователь ощутит большую задержку;

❑ `cache_peer_domain хост домен` — разрешить для соседнего прокси-сервера, расположенного по адресу `хост`, работу только с указанными доменами. Например, следующая строка позволит обращаться к соседнему прокси-серверу с адресом 192.168.2.20 (который должен быть описан директивой `cache_peer`) только за тем, что относится к домену `.com`:

```
cache_peer_domain 192.168.2.20 .com
```

Все остальные запросы не будут направляться на указанный соседний сервер, чтобы не перегружать его лишней работой. С помощью этого параметра можно настроить в сети несколько прокси-серверов, где каждый будет отвечать за свой домен.

9.3.6. Дополнительные директивы

Рассмотрим представляющие для нас ценность оставшиеся параметры, которые я не смог отнести к определенным категориям:

❑ `redirect_rewrites_host_header параметр` — разрешение (`on`) или запрет (`off`) на изменение поля `Host` в заголовках запросов. Если изменение разрешено, то сервер работает в анонимном режиме, иначе — в прозрачном. Анонимный режим требует дополнительных затрат, но позволяет использовать всего один IP-адрес для внешних соединений в сети любого размера. Прозрачный режим работает быстрее, но каждый компьютер должен иметь собственный IP-адрес, подходящий для работы с Интернетом;

- ❑ `redirector_access` список — перечень запросов, проходящих через перенаправитель (`redirector`). Перенаправитель — это небольшая программа, которая просматривает запрашиваемые URL и, возможно, заменяет их. Этот механизм позволяет реализовать довольно эффективный фильтр, запрещающий, например, просмотр порнографии. Более подробную информацию можно найти в *разд. 9.5.6* и в документации по `squid`. По умолчанию перенаправителю отправляются все запросы;
- ❑ `cache_mgr_email` — адрес E-mail, на который будет послано письмо в случае возникновения проблем с работой прокси-сервера;
- ❑ `append_domain` домен — домен по умолчанию. Например, чаще всего пользователи работают с адресами домена **.com**. Вполне логичным будет указать в директиве именно его (`append_domain .com`). Если пользователь введет адрес, например `redhat`, то `squid` сам добавит имя домена и направит на сайт **redhat.com**;
- ❑ `smtp_port n` — номер порта, на котором нужно ожидать SMTP-запросы для отправки сообщений. Конечно же, SMTP — это такой протокол, который не требует кэширования, и работа через прокси-сервер не сэкономит трафик, но возможность может оказаться полезной, если нельзя устанавливать шлюз, а разрешен только прокси-сервер;
- ❑ `offline_mode` параметр — режим работы. Если параметр равен `on`, то `squid` станет взаимодействовать только с кэшем, и обращений к Интернету не будет. Если запрашиваемой страницы в кэше нет, то пользователь увидит ошибку. Чтобы `squid` мог обращаться к Интернету, необходимо установить параметр `off` (установлено по умолчанию).

9.4. Права доступа к squid

Это самая болезненная тема для любого администратора. Да, и в `squid` тоже есть права доступа, и они также описываются в конфигурационном файле `/etc/squid/squid.conf`. Но мы рассматриваем права отдельно, потому что основной упор делаем на безопасность. Именно поэтому этой теме отведен отдельный раздел.

9.4.1. Список контроля доступа

Первое, с чем нам предстоит познакомиться, — это ACL (Access Control List, список контроля доступа), который предоставляет большие возможности для дальнейшей настройки прав доступа к сайтам. С помощью списка имен вы как бы группируете действия или пользователей. Используйте для этого следующую директиву:

```
acl имя тип параметр
```

У этой директивы три параметра:

- ❑ `имя` — может быть любым, но лучше всего, если оно будет описывать выполняемые действия;

- ❑ параметр — задает шаблон или строку, смысл которой зависит от типа записи (второй аргумент);
- ❑ тип — может принимать следующие значения: `src`, `dst`, `srcdomain`, `dstdomain`, `url_pattern`, `urlpath_pattern`, `time`, `port`, `proto`, `proxy_auth`, `method`, `browser`, `user`. Рассмотрим основные типы, которые вам пригодятся при формировании последнего параметра (шаблона):
- `src` — задаются IP-адреса пользователей;
 - `dst` — указываются адреса серверов;
 - `port` — определяется номер порта;
 - `proto` — описывается список протоколов (через пробел);
 - `method` — указывается используемый метод, например: `POST`, `GET` и т. д.;
 - `proxy_auth` — определяется список имен пользователей, значения разделяются пробелами. В качестве имени можно использовать `REQUIRED`, чтобы принимались любые действительные для сервера имена (`acl password proxy_auth REQUIRED`);
 - `url_regex` — устанавливается URL или регулярное выражение для URL;
 - `time` — задается время в формате дни `h1:m1-h2:m2`. С помощью такой записи можно ограничить доступ только определенными днями недели и обусловленным временем. В качестве дней недели можно указывать: `s` — Sunday (воскресенье), `m` — Monday (понедельник), `t` — Tuesday (вторник), `w` — Wednesday (среда), `h` — Thursday (четверг), `f` — Friday (пятница), `a` — Saturday (суббота).

В файле конфигурации уже описано несколько правил, которые готовы к использованию и в большинстве случаев не требуют вмешательства. Этот список прав доступа, необходимых для работы прокси-сервера, приведен в листинге 9.1.

Листинг 9.1. Список acl-правил, описанных по умолчанию в конфигурационном файле `/etc/squid/squid.conf`

```
acl all src 0.0.0.0/0.0.0.0
acl manager proto cache_object
acl localhost src 127.0.0.1/255.255.255.255
acl SSL_ports port 443 563
acl Safe_ports port 80          # http
acl Safe_ports port 21          # ftp
acl Safe_ports port 443 563     # https, snews
acl Safe_ports port 70          # gopher
acl Safe_ports port 210         # wais
acl Safe_ports port 1025-65535  # unregistered ports
acl Safe_ports port 280         # http-mgmt
acl Safe_ports port 488         # gss-http
```

```
acl Safe_ports port 591          # filemaker
acl Safe_ports port 777          # multiling http
acl CONNECT method CONNECT
```

Рассмотрим первую строку. Здесь задается ACL с именем `all`. Поскольку использован тип шаблона `src`, то этому списку принадлежат пользователи, у которых IP-адрес соответствует `0.0.0.0/0.0.0.0`, т. е. все пользователи.

Следующая строка создает ACL с именем `manager`. Она определяет доступ к протоколу, потому что тип записи `proto`, а последний параметр задает протокол — `cache_object`. И так далее.

Давайте попробуем задать свою запись ACL. Допустим, в вашей сети есть 10 компьютеров с адресами от `192.168.8.1` до `192.168.8.10` (маска подсети `255.255.255.0`), которым разрешен доступ к Интернету. Значит, всем остальным нужно запретить.

Уже при создании списка вы должны отталкиваться от идеи, что изначально доступ запрещен всем, и позволять его только тем, кому это действительно нужно. Итак, строка для всех у нас уже есть, и ее имя `all`. Для списка из 10 компьютеров создадим запись с именем `AllowUsers`, и ее описание будет следующим:

```
acl AllowUsers src 192.168.8.1-192.168.8.10/255.255.255.0
```

Эта запись относится к типу `src`, значит, сюда включаются все компьютеры с адресами, указанными в качестве последнего параметра.

9.4.2. Определение прав

После создания списков можно указать права доступа для каждого из них с помощью следующих команд:

- `http_access разрешение имя` — определяет права доступа по протоколу HTTP. В качестве параметра `разрешение` можно указывать `allow` (доступ разрешен) или `deny` (доступ запрещен). Последний аргумент `имя` — это имя ACL-записи. В следующем примере запрещается доступ по протоколу HTTP всем пользователям, кроме указанных в ACL-записи `AllowUsers`:

```
http_access deny all
http_access allow AllowUsers
```

Указав права доступа для списка `AllowUsers`, мы одной строкой даем разрешение для всех компьютеров, входящих в этот ACL. Таким образом, нет необходимости прописывать права каждому компьютеру в отдельности. Это значительно облегчает жизнь администраторов в больших сетях.

В предыдущем примере (см. *разд. 9.4.1*) мы описали список `AllowUsers` как список компьютеров с IP-адресами из диапазона `192.168.8.1–192.168.8.10`. Если к прокси-серверу обратится компьютер с другим адресом, то в доступе ему будет отказано;

- ❑ `icp_access` разрешение имя — описывает разрешение доступа к прокси-серверу по протоколу ICP. По умолчанию доступ запрещен для всех:

```
icp_access deny all
```

- ❑ `miss_access` разрешение имя — описывает разрешение на получение ответа MISSES. В следующем примере только локальным пользователям дано право получать ответ MISSES, а все остальные могут принимать только HITS:

```
acl localclients src 172.16.0.0/16
miss_access allow localclients
miss_access deny !localclients
```

9.4.3. Аутентификация

Защита по IP-адресу не гарантирует от его подделки злоумышленником. К тому же остается вероятность, что кто-то получит физический доступ к компьютеру, имеющему доступ во Всемирную сеть, и сделает нечто неразрешенное.

Мне довелось работать в фирме, где каждому сотруднику было разрешено скачивать из сети определенное количество мегабайт. Проверка проходила через IP-адрес, и при превышении лимита руководство требовало от работника покрыть расходы за сверхнормативный трафик. Это нормальная ситуация, потому что работодатель не должен оплачивать бессмысленные прогулки сотрудника в Интернете. На работу приходят, чтобы выполнять свои обязанности, а не подыскивать обои для собственного компьютера.

Однажды у некоторых работников оказалось большое превышение трафика. Вроде бы ничего удивительного, но настораживало то, что эти товарищи были в отпусках. Кто-то подделывал чужие адреса и использовал служебный Интернет в своих целях.

Чтобы вы не столкнулись с подобной ситуацией, необходимо привязываться не только к IP-адресу, но и строить дополнительную защиту через проверку имени пользователя и пароля. Для аутентификации необходимо определить следующие директивы:

- ❑ `authenticate_program` программа файл — задает программу аутентификации (по умолчанию не используется) и файл паролей. Если вы хотите применить традиционную программу аутентификации, то можно указать следующую строку:

```
authenticate_program /usr/lib/squid/ncsa_auth/usr/etc/passwd
```

Путь к программе `ncsa_auth` в вашей системе может быть иным.

Чтобы использовать возможности аутентификации прокси-сервера, у вас должна присутствовать хотя бы одна ACL-запись типа `proxy_auth`;

- ❑ `authenticate_children n` — определяет количество параллельно работающих процессов аутентификации. Один процесс не может осуществлять проверку нескольких клиентов, поэтому если один пользователь проходит аутентификацию, то другие не смогут получить доступ к Интернету через сервер `squid`;

- ❑ `authenticate_ttl n hour` — срок (в часах) хранения в кэше результата аутентификации. В течение этого времени пользователь может работать без повторной проверки. По умолчанию установлен 1 час, но если введен неправильный пароль, то запись удаляется из кэша;
- ❑ `authenticate_ip_ttl 0 second` — связывает IP-адрес с аутентификацией. Необходимо установить 0, чтобы пользователи не могли воспользоваться одним и тем же паролем с разных IP-адресов. Некоторые пользователи считают, что можно делиться паролем с друзьями, но не стоит им это разрешать, потому что за раздачу паролей должен отвечать только администратор.

Если в вашей сети есть dialup-пользователи, подключающиеся с помощью модема, то это значение можно увеличить до 60 секунд, чтобы при разрыве связи была возможность перезвонить. Но обычно при dialup-подключении используются динамические IP-адреса, которые выдаются при каждом соединении, и нет гарантии, что после повторного звонка адрес сохранится;

- ❑ `authenticate_ip_ttl_is_strict параметр` — если параметр равен `on`, то доступ с других IP-адресов запрещен, пока время, указанное в `authenticate_ip_ttl`, не выйдет.

ВНИМАНИЕ!

Аутентификация не работает, если squid настроен на работу в прозрачном режиме.

9.5. Некоторые нюансы работы со squid

Сейчас нам предстоит немного поговорить о некоторых вопросах безопасности сервиса squid и дополнительных возможностях, которые могут ускорить работу в Интернете.

9.5.1. Безопасность сервиса

Когда я впервые знакомился с документацией на squid, то мне очень понравились следующие два параметра: `cache_effective_user` и `cache_effective_group`. Если squid запущен от имени администратора `root`, то идентификаторы пользователя и группы будут заменены на указанные в этих параметрах. По умолчанию установлено значение идентификатора squid и для пользователя, и для группы:

```
cache_effective_user squid
cache_effective_group squid
```

Таким образом, squid не будет работать с правами `root`, и при попытке сделать это сервис сам понизит свои права до squid. Не стоит вмешиваться. Сервису squid не имеет смысла давать большего, потому что ему достаточно прав только на работу с каталогом кэша.

9.5.2. Ускорение сайта

Сервис squid может ускорить работу определенного сайта, функционируя как httpd-акселератор. Для этого необходимо указать как минимум три параметра: адрес ускоряемого сервера, который надо кэшировать, его порт и атрибуты сервера, который надо ускорять. Это делается с помощью следующих директив:

- ☐ `httpd_accel_host` адрес — адрес реального сервера;
- ☐ `httpd_accel_port` порт — порт веб-сервера. Чаще всего это порт по умолчанию (он равен 80), если не указан другой;
- ☐ `httpd_accel_uses_host_header` параметр — заголовок HTTP включает в себя поле `Host`. Сервер squid не проверяет `Host`, и это может оказаться большой дырой в безопасности. Разработчики рекомендуют отключать эту директиву, указав в качестве параметра значение `off`. Включать ее необходимо, если squid работает в прозрачном режиме;
- ☐ `httpd_accel_with_proxy` параметр — флаг использования кэширования страницы для дополнительного повышения скорости (`on/off`).

9.5.3. Маленький секрет поля *User Agent*

Многие статистические системы не учитывают или не пускают к себе пользователей, запросы которых содержат пустое значение в поле `User Agent`. Именно так определяется, что вы работаете через прокси-сервер.

Опять случай из собственной практики. Я снова вспоминаю фирму, где мне пришлось работать четыре года и где защита была организована по IP-адресу. Мой отдел занимался автоматизацией производства, и в нем работали электронщики, а я был единственный программист и администратор в одном лице. Доступ в Интернет был разрешен только мне, начальнику отдела и его заместителю. Через несколько часов доступ имели все сотрудники отдела. Как это происходило? Решение очень простое — я поставил на свой компьютер прокси-сервер, к которому могли подключаться без аутентификации мои сослуживцы, а он уже переправлял эти запросы корпоративному прокси-серверу. Поскольку все запросы шли от меня, то главный прокси-сервер не возражал.

На первый взгляд решение идеальное, но тут есть один изъян. Да, это поле `User Agent`, которое становится пустым при прохождении пакетов через мой squid-сервис. Но поле можно задать вручную в конфигурационном файле. Для этого существует директива `fake_user_agent`. Например, следующая строка может эмулировать запросы, как будто они идут от браузера Netscape:

```
fake_user_agent Netscape/1.0 (CP/M; 8-bit)
```

9.5.4. Защита сети

Сервис squid может быть как средством защиты сети, так и орудием проникновения хакера в сеть. Чтобы внешние пользователи не могли задействовать прокси-сервер

для подключения к компьютерам локальной сети, необходимо добавить в конфигурационный файл следующие строки:

```
tcp_incoming_address  внутренний_адрес
tcp_outgiong_address  внешний_адрес
udp_incoming_address  внутренний_адрес
udp_outgiong_address  внешний_адрес
```

Здесь `внутренний_адрес` — это адрес компьютера с установленным squid, сетевое соединение которого направлено на вашу локальную сеть, а `внешний_адрес` — это адрес сетевого соединения, направленного в Интернет. Если неправильно указать адреса, то хакер сможет подключаться к компьютерам локальной сети, находясь за ее пределами. Вот пример ошибочного конфигурирования squid-сервиса:

```
tcp_incoming_address  внешний_адрес
tcp_outgiong_address  внутренний_адрес
udp_incoming_address  внешний_адрес
udp_outgiong_address  внутренний_адрес
```

9.5.5. Борьба с баннерами и всплывающими окнами

В фирме, где я работал, появился новый сотрудник, и в первую неделю мы ощутили увеличение трафика. Это бывает со всеми, потому что любой новый пользователь Интернета начинает смотреть все страницы подряд. Со временем интерес стихает, и трафик понижается.

Мы уже говорили о том, что на любом сайте большую часть трафика отнимает графика. В большинстве браузеров отображение картинок можно отключить, но после этого путешествие окажется не очень удобным. Некоторые сайты без графики теряют информативность, и с ними становится сложнее работать, поэтому отказаться совсем от этого режима невозможно.

Но есть графика, которая надоедает, раздражает и не несет полезной информации, а главное, от нее можно избавиться. Я говорю про баннеры. Давайте рассмотрим, как их можно отключить еще на уровне прокси-сервера. Для этого сначала добавим в файл `squid.conf` следующие правила:

```
acl banners_regex url_regex "/usr/etc/banners_regex"
acl banners_path_regex urlpath_regex "/usr/etc/banners_path_regex"
acl banners_exclusion url_regex "/usr/etc/banners_exclusion"
```

Первая строка создает ACL-список с именем `banners_regex` типа `url_regex`, который позволяет сравнивать полный URL-адрес. В последнем параметре определен файл `/usr/etc/banners_regex`, в котором будут указываться нужные адреса. Нас интересуют URL баннерных систем, и вы можете поместить их в этот файл.

Вторая строка создает ACL-список с именем `banners_path_regex` типа `urlpath_regex`. В последнем ее параметре снова указан файл `/usr/etc/banners_path_regex`, в котором вы должны описывать пути URL, которые впоследствии мы запретим.

Третья строка схожа с первой, но имеет имя `banners_exclusion` и связана с файлом `/usr/etc/banners_exclusion`. В первых двух файлах вы должны описывать пути или

шаблоны, по которым потом будут обрезаться баннеры. Но бывают случаи, когда можно промахнуться и отсечь вполне полезную информацию. Если найден ошибочный путь, то его можно записать в этот файл, и баннер будет загружен.

Теперь добавляем еще две строки после описания ACL-записей:

```
http_access deny banners_path_regex !banners_exclusion
http_access deny banners_regex !banners_exclusion
```

Обе директивы имеют один и тот же смысл — запрещается загрузка по адресам, прописанным в списке `banners_path_regex` или `banners_regex`, если адрес не входит в исключение, описанное в файле ACL-списка `banners_exclusion`.

Рассмотрим фрагмент содержимого файла `/usr/etc/banners_regex`:

```
^http://members.tripod.com/adm/popup/.+html
^http://www.geocities.com/ad_container/pop.html
```

Напоминаю, что в этом файле находятся URL-пути для сравнения, и все адреса, которые им соответствуют, будут отфильтрованы.

Шаблону из первой строки соответствует, например, адрес **`http://members.tripod.com/adm/popup/popup.html`**, так что он не будет загружен. Да, так просто. И пользователи больше не увидят всплывающие окна с сайта **`tripod.com`**. Если вы знакомы с регулярными выражениями, то сможете создать подобные записи для любой баннерной системы и обрезать самые замысловатые пути надоедливых картинок. Я не буду затрагивать регулярные выражения, потому что это тема столь велика, что требует отдельной книги.

При борьбе с баннерами будьте готовы к тому, что «обрезание» не всегда помогает, — всплывающие окна могут снова появиться через определенное время. Это связано с тем, что баннеры — просто реклама и позволяют зарабатывать деньги на существование сайта. Особо одаренные администраторы ищут любые возможные пути, чтобы ваша система не смогла избавиться от рекламы. Для этого они постоянно изменяют адреса, в результате чего регулярное выражение перестает срабатывать.

9.5.6. Подмена баннера

Пока что мы запретили загрузку баннеров или всплывающих окон. Но после этого веб-страницы перестают быть привлекательными. Чтобы этого не произошло, можно заменять баннеры на свои картинки, которые хранятся на сервере, и отпадет необходимость грузить их из Интернета.

Для решения этой задачи очень хорошо подходит перенаправитель (redirector). Для сервиса `squid` — это внешняя программа, которая подменяет адреса. Например, если сайту необходимо загрузить баннер и ваша программа смогла определить такую попытку, то `redirector` подменит адрес и вместо баннера загрузит то, что укажете вы.

Есть только одна проблема — в ОС нет и не может быть такой готовой программы. Ее необходимо написать. Для этого подойдет любой язык программирования, а я

покажу вам пример, реализованный на языке Perl. Если вы умеете программировать на этом языке, то способ с `redirector` понравится вам больше, чем простой запрет через ACL.

Пример классической программы `redirector` можно увидеть в листинге 9.2. Я постарался максимально упростить его, чтобы вам легче было адаптировать сценарий под свои задачи.

Листинг 9.2. Сценарий на языке Perl для подмены баннеров и закрытия всплывающих окон

```
#!/usr/bin/perl

$| = 1;

# Укажите URL на вашем веб-сервере, где лежат картинки
$YOURSITE = 'http://yourserver.com/squid';
$LOG = '/usr/etc/redirectlog';
$LAZY_WRITE = 1;

if ($LOG) {
    open LOG, ">> $LOG";
    unless ($LAZY_WRITE)
    {
        select LOG ;
        $| = 1 ;
        select STDOUT;
    }
}

@b468_60 = qw (
    www\.sitename\.com/cgi/
    # Добавьте сюда описания URL-адресов с баннерами
    # размером 468x60
);

@b100_100= qw (
    www\.sitename\.com/cgi/
    # Добавьте сюда описания URL-адресов с баннерами
    # размером 100x100
);

@various = qw (
    www\.sitename\.com/cgi/
    # Добавьте сюда описания URL-адресов с нестандартными
    # размерами баннера
);
```

```

@popup_window = qw (
    ^http://members.tripod.com/adm/popup/.+html
    ^http://www.geocities.com/ad_container/pop\.html
    ^http://www.geocities.com/toto\?
    # Добавьте сюда описания URL-адресов, с которых
    # выскакивают всплывающие окна
);

# Описание расположения картинок
$b468_60      = "$YOURSITE/468_60.gif";
$b100_100     = "$YOURSITE/100_100.gif";
$svarious     = "$YOURSITE/empty.gif";
$closewindow = "$YOURSITE/close.htm";

while (<>)
{
    ($url, $who, $ident, $method) = /^(\\S+) (\\S+) (\\S+) (\\S+)$/;
    $prev = $url;

    # Проверка баннера 468x60
    $url = $b468_60 if grep $url =~ m%$_%, @b468_60;

    # Проверка баннера 100x100
    $url = $b100_100 if grep $url =~ m%$_%, @b100_100;

    # Проверка баннера произвольного размера
    $url = $svarious if grep $url =~ m%$_%, @various;

    # Всплывающее окно
    $url = $closewindow if grep $url =~ m%$_%, @popup_window;

    # Отдельный сайт, не внесенный в список в начале файла
    $url = "$YOURSITE/empty.gif" if $url =~ m%hitbox.com/Hitbox\?%;

    if ($LOG and $url ne $prev)
    {
        my ($sec, $min, $hour, $mday, $mon, $year) = localtime;
        printf LOG "%2d.%02d.%2d %2d:%02d:%04d: %s\r\n",
            $mday, $mon + 1, $year + 1900, $hour, $min, $sec,
            "$who $prev > $url";
    }

    print "$url $who $ident $method\n";
}

close LOG if $LOG;

```

Код в листинге 9.2 в достаточной степени снабжен комментариями. Если у вас есть опыт программирования на Perl, то дальнейшие действия вы выполните без проблем.

Сохраните эту программу в файле `/usr/etc/redirector` и установите для squid права на его исполнение. После этого добавьте в файл `squid.conf` следующую строку:

```
redirect_program /usr/local/etc/squid/redirector
```

Чтобы программа заработала, создайте на своем веб-сервере файлы со следующими именами:

- ☐ `468_60.gif` — картинка размером 468×60;
- ☐ `100_100.gif` — картинка размером 100×100;
- ☐ `empty.gif` — картинка, которая будет заменять нестандартные баннеры. Лучше всего ее сделать размером 1×1 пиксел, чтобы она не испортила дизайн сайта;
- ☐ `close.htm` — файл HTML, который будет закрывать всплывающие окна. В нем нужно поместить всего лишь функцию, которая будет закрывать окно. Для этого служит функция JavaScript `window.close()`. Пример содержимого файла показан в листинге 9.3.

Все эти файлы должны лежать на веб-сервере в одном каталоге. Не забудьте в сценарии (в переменной `$YOURSITE`) указать правильный путь к этому каталогу.

Листинг 9.3. Пример JavaScript-файла, закрывающего всплывающее окно

```
<html>
<head>
<script language="JavaScript">
<!--
    window.close();
//-->
</script>
</head>
<body>
</body>
</html>
```

9.5.7. Борьба с запрещенными сайтами

Недавно я разговаривал с одним своим знакомым, и мне понравилось его определение Интернета: сеть создана и живет порнографией. Я не совсем в этом уверен, но мне кажется, что он прав хотя бы в том, что трафик с сайтов с интим-содержимым наиболее высок (если не считать службу обновления Microsoft, откуда пользователи скачивают патчи для программ этой компании ☺).

Ни один работодатель не обрадуется, если его сотрудники в рабочее время будут посещать сайты с запрещенным контентом (это не только бесполезная трата трафи-

ка, но и другие непроизводительные расходы). Родители тоже не хотят, чтобы их дети посещали подобные сайты, поэтому стремятся оградить их от этого зрелища. Это я говорю как отец двоих детей.

Порносайты легко можно запретить с помощью таких же методов, как мы использовали для баннеров. Например, можно отключить любые сайты, в адресе которых есть слово «sex». Но нельзя забывать, что могут быть исключения. К примеру, адрес может содержать текст «GasExpo». Обратите внимание, что выделенные буквы создают слово «sex». И случаи, когда пользователи попадают отнюдь не на сайт выставки по газовому оборудованию, вполне реальны.

Создавать списки запрещенных сайтов достаточно сложно, но можно. В настоящее время в зоне **com** большинство сайтов эротической направленности закрылись, и они обживают другие домены, принадлежащие маленьким государствам. Существуют домены, которые на 90% состоят из сайтов индустрии развлечений для взрослых. Вот их можно обрезать полностью.

9.5.8. Ограничение канала

При организации доступа в Интернет очень часто требуется обеспечить отдельным пользователям бóльшую скорость подключения. Как это сделать, когда по умолчанию все равноправны и могут работать на максимально доступной в текущий момент скорости? Для этого нужно определиться с приоритетами. Для некоторых пользователей должен быть зарезервирован высокоскоростной канал связи. Но нельзя повысить скорость одному человеку без ущерба остальным.

Если кому-то требуется полоса гарантированной пропускной способности для работы с приложениями, требующими высокой скорости обмена (например, для проведения презентаций), вы должны зарезервировать для них более мощный, чем для остальных, канал.

Ограничение внешнего канала достаточно легко выполнить с помощью squid. Директивы, которые нужно использовать, можно увидеть в следующем примере:

```
delay_pools 3
delay_class 1 1
delay_class 2 2
delay_class 3 1
delay_parameters 1 256000/256000
delay_access 1 deny all
delay_access 1 allow admins
delay_parameters 2 256000/256000 4000/8000
delay_access 2 allow all
delay_access 2 deny admins
delay_parameters 3 64000/64000
delay_access 3 deny all
delay_access 3 allow bigboss
```

Этот код нужно добавить в файл конфигурации `/etc/squid/squid.conf` после комментария:

```
# DELAY POOL PARAMETERS (all require DELAY_POOLS compilation option)
# -----
```

Большинство параметров здесь заданы по умолчанию, и их следует заменить в соответствии с существующей у вас ситуацией.

Давайте подробно рассмотрим такую конфигурацию. Для начала нужно определить, сколько у вас будет пулов (правил, описывающих скорость доступа). Для этого используется директива `delay_pools n`, где `n` — это количество пулов. По умолчанию `n` равно нулю, и нет никаких ограничений. Мы создадим три пула, поэтому в примере указано число 3.

После этого нужно определить, к какому классу относится пул. Это делается с помощью директивы `delay_class n c`, где `n` — это порядковый номер, а `c` — номер класса. Каждая строка с директивой `delay_class` должна иметь свой порядковый номер, который начинается с 1. В нашем примере три строки, поэтому в первой параметр `n` равен 1, во второй — 2, а в третьей — 3.

Номеров класса (второй параметр) может быть три:

- 1 — ограничение канала происходит для всей сети. Например, у вас внешний канал 256 Кбит/с — вы можете его уменьшить для всех до 64 Кбит/с;
- 2 — сузить можно общий канал, а также для каждого пользователя индивидуально. В этом случае можно понизить общий канал до 64 Кбит/с, а каждому пользователю — до 4 Кбит/с;
- 3 — ограничивать можно общий канал, индивидуально и для каждой сети в отдельности. Например, если скорость канала равна 256 Кбит/с, а в вашей сети работают 4 подсети, то каждой из них можно выделить по 64 Кбит/с, чтобы равномерно разделить нагрузку.

В нашем примере мы используем пулы классов 1, 2 и снова 1. Я специально расположил их не последовательно, чтобы пример был более наглядным.

Теперь описываем параметры скорости доступа. Это делается с помощью следующей директивы:

```
delay_parameters пул скорость_канала скорость_сети индивидуально
```

Здесь `пул` — это номер пула, скорость которого мы хотим описать. Так, в нашем примере следующая строка описывает скорость для первого пула:

```
delay_parameters 1 256000/256000
```

Количество параметров зависит от класса используемого пула. Так как пул 1 имеет класс 1 (`delay_class 1 1`), у которого можно ограничивать только канал полностью, в директиве используется единственный параметр — скорость канала: `256000/256000`. Как можно видеть, он формируется в виде двух чисел, разделенных знаком `/`. До косой черты указывается скорость, с которой будут скачиваться данные из сети, а после нее — размер пула, т. е. емкость, которую можно наполнить полученными из сети данными.

Обратите внимание, что скорость указывается именно в байтах, а характеристика модема задается в битах в секунду. Если в качестве скорости указать значение -1, то никаких ограничений не будет.

После определения параметров для первого пула нужно установить права доступа к нему. Это делается директивой `delay_access`, которая имеет следующий вид:

```
delay_access пул доступ acl
```

Первый параметр — это снова номер пула. Потом указывается доступ `allow` или `deny`, и последним идет имя ACL-списка.

В нашем примере для первого пула записаны две строки:

```
delay_access 1 deny all
delay_access 1 allow admins
```

Сначала мы запрещаем доступ для всех, а потом разрешаем работать на заданной скорости только ACL-списку `admins`. Подразумевается, что в этот список входят администраторы.

Далее идет описание скорости и прав доступа для второго пула:

```
delay_parameters 2 256000/256000 4000/8000
delay_access 2 allow all
delay_access 2 deny admins
```

Так как второй пул относится ко 2-му классу, то здесь нужно указать общую скорость (256 000 байтов в секунду) и скорость доступа каждого компьютера в отдельности — 4000 байтов в секунду. На такой скорости будут работать все пользователи в сети, кроме администраторов.

Если вы установите такие правила в какой-либо организации, то могут возникнуть проблемы с руководством, потому что директор тоже попадет в список `all` и будет вынужден работать на скорости 4000 байтов в секунду. Я думаю, что его это не устроит, и для него мы сделаем отдельную запись:

```
delay_parameters 3 64000/64000
delay_access 3 deny all
delay_access 3 allow bigboss
```

С помощью ограничения пропускной способности канала можно запретить загрузку мультимедиафайлов в рабочее время. В следующем примере мы разрешаем быстро читать веб-странички, но уменьшаем скорость загрузки других медиафайлов (листинг 9.4).

Листинг 9.4. Ограничение скорости загрузки медиафайлов в рабочее время

```
# ACL-список, описывающий сеть
acl fullspeed url_regex -i 192.168.1
# ACL-список, описывающий медиафайлы, для которых необходимо
# понизить скорость
acl mediaspeed url_regex -i ftp .exe .mp3 .avi .mpeg .iso .wav
```

```
# Время, в течение которого будет действовать ограничение на скорость
# загрузки медиафайлов
acl day time 08:00-18:59

# Нам нужно два пула второго класса
delay_pools 2
delay_class 1 2
delay_class 2 2
# Первый пул не имеет ограничений для всех
delay_parameters 1 -1/-1 -1/-1
delay_access 1 allow fullspeed

# Второй пул ограничивает доступ в дневное время
delay_parameters 2 4000/100000 4000/100000
delay_access 2 allow day
delay_access 2 deny !day
delay_access 2 allow mediaspeed
```

Я постарался снабдить листинг подробными комментариями, чтобы вы могли с ним разобраться без дополнительных пояснений. Хочу только заметить, что скорость понижается для всех. Если вы хотите разрешить определенным пользователям работать на полной скорости, можно внести их в список (например, `allowfull`) и добавить в конец листинга следующую строку:

```
delay_access 2 deny !allowfull
```

9.6. Защита прокси-сервера: squidGuard

Использование прокси-сервера довольно опасно, поэтому squid очень часто применяют совместно с дополнительной защитой в виде программы squidGuard. В качестве возможностей этого стражника можно выделить опознавание пользователей по имени или адресу, фильтрацию запросов, замену баннеров и т. д.

9.6.1. Установка

Установка программы весьма интересна. Скачайте исходные коды с сайта www.squidguard.org. Скачав исходники, разархивируйте их. Это можно сделать даже графической утилитой, которая работает в Ubuntu как файловый менеджер. Для простоты разархивируйте их в свой домашний каталог. Запустите терминал и перейдите внутрь созданного при разархивации каталога с программой:

```
cd ~/squidGuard-X.X
```

В нашем случае `X.X` — это номер версии squidGuard. Запустите отсюда команду конфигурации:

```
./configure
```

По экрану побегут строки, сообщающие о ходе проверки и конфигурирования программы. Если выполнение прервалось с ошибкой, то, вероятнее всего, в вашей системе отсутствует BerkeleyDB. У меня в Ubuntu так и было, поэтому сразу расскажу, как станцевать танец с бубнами для установки этого зверя. При помощи поисковика находим архив BerkeleyDB, скачиваем его и разархивируем. Лучше опять разархивировать в домашнюю папку — все равно потом все это можно будет удалить. Теперь переходим в папку:

```
cd ~/db-X.X.X/build-unix
```

Буквы *x* снова означают здесь номер версии. Обратите внимание, что мы переходим в подкаталог `build-unix`. Это очень важно — мы устанавливаем библиотеку работы с базой в Linux, которая относится к системам UNIX, поэтому компиляция и установка должны происходить именно отсюда. В этом каталоге выполняем три команды:

```
../dist/configure  
make  
sudo make install
```

Первая команда конфигурирует программу, вторая ее компилирует, а третья — устанавливает. Обратите внимание, что перед командой установки стоит `sudo`. Дело в том, что установка будет происходить в каталог `/usr/local`, к которому у простых смертных учетных записей доступа нет. Если не указать `sudo`, то установка завершится ошибкой прав доступа.

Теперь, когда библиотека базы данных установлена, можно возвращаться в каталог `squidGuard`. Только не спешите запускать конфигурацию снова. С вероятностью 99% она не пройдет, потому что конфигуратор ищет библиотеку BerkeleyDB в папке `/usr/local/BerkeleyDB`, но она устанавливается в папку `/usr/local/BerkeleyDB.X.X`, т. е. в конец имени папки добавляется номер версии. Выполните команду:

```
ls /usr/local
```

Это позволит увидеть содержимое папки `/usr/local` прямо из текущей папки стражника. Запомните имя и выполняйте команду конфигурации так:

```
./configure --with-db=/usr/local/BerkeleyDB.X.X
```

При этом концевые `X.X` заменяем на свой номер версии. Теперь конфигурация должна пройти успешно. Если нет, то в этом случае все претензии к разработчикам, но я с другими проблемами не встречался.

После конфигурации выполняем команды сборки и установки:

```
make  
sudo make install
```

Установка снова выполняется от имени администратора, потому что опять происходит копирование в защищенную область `/usr/local`.

9.6.2. Настройка

Каталог по умолчанию, в котором можно найти установленный нами стражник, — `/usr/local/squidGuard`. Основные его настройки находятся в файле `squidGuard.conf`, расположенном в том же каталоге. Первое, что вы увидите в этом файле, — это следующие две строки:

```
dbhome /usr/local/squidGuard/db
logdir /usr/local/squidGuard/logs
```

Первый параметр (`dbhome`) задает путь к базам данных, где будут храниться списки доступа. Второй параметр (`logdir`) — это путь к журналу. На сайте www.squidguard.org в разделе **Blacklists** можно скачать примеры списков доступа с вредными сайтами. Их нужно разархивировать в каталог баз данных, т. е. в тот каталог, что указан в параметре `dbhome`.

Теперь мы можем в этом же файле задать права доступа. Причем права доступа могут распределяться даже по времени и по пользователям. Отдельным пользователям можно дать возможность смотреть все, а кому-то (например, детям) запретить смотреть сайты для взрослых.

Для создания времени используется параметр `time`. Например, можно создать период, который будет отображать рабочее время. Для этого в файле `squidGuard.conf` пишем:

```
time worktime {
    weekly * 9:00-17:00          # Рабочее время
    date 2010.02.27 9:00-17:00   # День, который случайно стал рабочим
}
```

После параметра `time` указывается имя периода. В фигурных скобках задаем периоды времени. Параметр `weekly` со звездочкой означает любой день недели. Если нужно определиться только с рабочими днями недели, то их можно указать так: `MTWTF`. После этого идет рабочее время, в которое и будет задействовано создаваемое правило. Параметр `date` позволяет задать определенную дату. В нашем случае я выбрал для примера субботу, 27 февраля 2010 года, которая постановлением правительства была назначена рабочим днем. Наше правило будет распространяться и на этот день.

Для задания адресов, к которым разрешен или запрещен доступ, служит директива `dest`, например:

```
dest porn {
    domainlist      porn/domains
    urllist          porn/urls
    expressionlist  porn/expressions
}
```

Как видно из примера, здесь указываются параметры `domainlist`, `urllist` и/или `expressionlist`, с помощью которых задаются имена файлов, содержащих списки доменов, URL или регулярных выражений. Пути задаются относительно каталога,

Затем права доступа можно подкорректировать следующим образом:

```
acl {  
    admins within worktime {  
        pass all  
    }  
    others within worktime {  
        pass    !adv !porn !warez all  
    }  
    else {  
        pass all  
    }  
    default {  
        pass    none  
        redirect http://localhost/block.html  
    }  
}
```

Теперь вы сможете обращаться к любым веб-сайтам. Чтобы изменения вступили в силу, нужно из каталога `/usr/local/bin` выполнить следующую команду:

```
squidGuard -C all
```

9.7. Шлюз в Интернет

Linux-машина может использоваться в качестве точки доступа к Интернету для целых сетей. В настоящее время большинство подключается к Интернету через маршрутизаторы. Именно маршрутизатор имеет подключение к Интернету, а все пользователи локальной сети подключаются к этому маршрутизатору, который с помощью NAT (Network Address Translation, преобразование сетевых адресов) позволяет большому количеству компьютеров видеть Интернет.

Linux тоже может работать в качестве шлюза во Всемирную сеть. Если у вашего компьютера есть два сетевых интерфейса, к одному из которых подключен Интернет, а к другому — локальная сеть, то этот компьютер сможет раздавать доступ в Интернет всем пользователям сети.

Итак, у нас есть два сетевых интерфейса: `eth0` (подключен к Интернету) и `eth1` (подключен к локальной сети). Все пользователи вашей сети подключены к сетевой карте `eth1`, и они хотят получить доступ к Интернету.

Для начала нужно убедиться, что включен форвардинг пакетов:

```
cat /etc/sysctl.conf | grep net.ipv4.ip_forward
```

В результате вы должны увидеть строку:

```
net.ipv4.ip_forward = 1
```

В Ubuntu по умолчанию вы увидите эту строку, но вначале будет стоять символ решетки, а это значит, что параметр не работает, а является комментарием. Нужно убрать символ решетки и включить форвардинг.

Чтобы включить форвардинг, можно выполнить команду:

```
sysctl -w net.ipv4.ip_forward=1
```

После перезагрузки эта конфигурация может быть потеряна. Чтобы сохранить ее, нужно изменить параметр `net.ipv4.ip_forward` в файле `/etc/sysctl.conf`. Откройте файл в любом редакторе и убедитесь, что параметр в нем существует и установлен в 1, и при этом строка не закрыта символом комментария.

Теперь нужно установить сам форвардинг данных из одной сети в другую, и тут отлично подходят возможности сетевого экрана. Для его настройки я решил использовать сервис `firewalld`, который мы устанавливали в *разд. 4.10.8*.

Первым делом устанавливаем локальную сеть в качестве доверенной:

```
firewall-cmd --permanent --zone=trusted --add-source=192.168.1.0/24
```

Здесь 192.168.1.0 — это IP-адрес локальной сети, который вам нужно будет заменить на собственный (в больших сетях часто встречаются адреса вида 10.x.x.x.). Число после слэша — это маска сети (число 24 соответствует маске 255.255.255.0).

Теперь привяжем сетевую карту `eth1` (в моем примере она смотрит на локальную сеть) к внутренней зоне:

```
sudo firewall-cmd --change-interface=eth1 --zone=internal --permanent
```

Интерфейс, который смотрит в Интернет (`eth0`), привязываем к внешней зоне:

```
firewall-cmd --change-interface=eth0 --zone=external --permanent
```

Теперь нужно включить маскарading для внешней сети:

```
firewall-cmd --zone=external --add-masquerade --permanent
```

Ну а теперь самое важное — правило, которое позволит направлять трафик в Интернет:

```
firewall-cmd --permanent --direct --passthrough ipv4 -t nat -I POSTROUTING -o eth0 -j MASQUERADE -s 192.168.1.0/24
```

Этой командой разрешается передавать трафик IPv4 с одного интерфейса на другой.

Чтобы изменения вступили в силу, нужно перезагрузить `firewalld`:

```
firewall-cmd --reload
```

Если я ничего не забыл, то теперь вы должны оказаться в Интернете. Опционально можно разрешить DNS-трафик:

```
firewall-cmd --zone=internal --add-service=dns --permanent
```




ГЛАВА 10

Передача файлов

Были времена, когда построение сети было делом дорогим, а Интернет — еще дороже, и для обмена файлами приходилось бегать с дискетами 5,25 или 3,5 дюйма. Если кто-то застал те времена, наверное, он вспоминает их с ужасом. Дискеты постоянно портились, и данные с них периодически переставали читаться. Благо если расстояние было небольшое и можно было повторить пробежку, но когда дистанция большая — потеря очень сильно отражалась на эмоциональном состоянии.

До сих пор в некоторые компьютеры вставляют дисководы для 3,5-дюймовых дискет, видимо, по привычке. Но уже трудно представить себе офис без полноценной сети, и в фирмах, где мне приходилось настраивать серверы, все компьютеры были обязательно подключены к локальной сети. В таких случаях уже нет надобности в дисководах, и администраторы их просто вынимают. Если у вас возник вопрос, зачем вытаскивать то, что может пригодиться, значит, вы забыли, что ничего лишнего в компьютере быть не должно. Это касается не только программ, но и компьютерного «железа».

Через дискеты хакеры легко могут унести секретную информацию, получив физический доступ к компьютеру. Да, много не унесешь, но все же... Мне известна фирма, которая содержала локальную сеть, отрезанную от внешнего мира, и считала себя неприступной. Но секретная информация утекла через все охраняемые системы с помощью простых 3,5-дюймовых дискет, потому что металлоискатели на них не срабатывали, — так были потеряны рынки сбыта, и только после этого все компьютеры фирмы лишили столь опасного устройства.

Сети позволяют избавиться от лишнего оборудования в компьютерах и сделать передачу данных быстрее и надежнее. Надо всего лишь настроить нужные протоколы и использовать кабельную проводку на полную мощность.

В настоящее время самым популярным протоколом обмена файлами является FTP (File Transfer Protocol, протокол передачи файлов). Он был разработан достаточно давно, но до сих пор не потерял своей актуальности, хотя некоторые его возможности оставляют желать лучшего.

10.1. Протокол FTP

Как мы уже говорили в *главе 6*, для работы по протоколу FTP требуется две составляющие: клиент и сервер. В качестве клиента в принципе можно использовать любой Telnet-клиент и, подключившись к порту 21 сервера, вручную передавать команды.

Если у вас нет FTP-клиента, то тестирование протокола можно осуществить даже через браузер — например, Internet Explorer или Firefox. Для этого в адресной строке браузера нужно набрать URL в формате: `ftp://имя:пароль@адрес`.

Например:

```
ftp://flenov:mypassword@ftp.my_server.com
```

или

```
ftp://flenov:mypassword@192.168.77.1
```

Протокол FTP работает сразу на двух портах: один служит для управления (пересылка команд), а другой — для передачи данных (файлов). Программа-клиент соединяется с портом 21 и начинает передавать команды. С этим портом соединяются все пользователи, и сервис, который прослушивает этот канал, работает одновременно с несколькими соединениями.

Когда клиент запрашивает данные, открывается еще одно соединение с конкретным пользователем, по которому передается файл. Это удобно с точки зрения программирования, но несподручно в плане администрирования — точнее сказать, конфигурирования сетевого экрана. Дело в том, что для передачи данных клиент может открыть абсолютно любой порт, и сервер не может предсказать его номер.

Большинство команд, применяемых в работе по протоколу FTP, схожи с теми директивами, которые вы используете в Linux для управления файлами. Это связано с тем, что во время разработки протокола основными сетевыми ОС являлись UNIX-системы. Это в наше время везде стоит Windows, а 50 лет назад все было иначе.

10.1.1. Команды протокола FTP

Давайте рассмотрим листинг 10.1. В нем приведен пример общения клиентской программы с FTP-сервером. Строки, в начале которых стоит знак `>`, были отправлены FTP-серверу, а остальные — это его ответы.

Листинг 10.1. Пример работы протокола FTP

```
< 220 Flenov Mikhail FTP Server
> USER Anonymous
< 331 Anonymous access allowed, send identity (e-mail name) as password.
> PASS your@mail.com
< 230 Anonymous user logged in.
> PWD
```

```
< 257 "/" is current directory.  
> TYPE A  
< 200 Type set to A.  
> PASV  
< 227 Entering Passive Mode (127,0,0,1,13,20).  
> LIST  
< 125 Data connection already open; Transfer starting.  
< 226 Transfer complete.
```

Самой первой строкой идет приглашение сервера FTP. Его мы получаем сразу же в ответ на соединение с портом 21. В этой строке чаще всего находится текстовое описание сервера, с которым произошло соединение, и его версия. В нашем случае здесь вместо конкретного названия вписано мое имя, но в реальном сервере при настройках по умолчанию будет видна примерно следующая строка:

```
220 flenovm.ru FTP server (Version wu-2.6.2-5) ready.
```

Для чего я изменил строку приветствия? Все очень просто — в ней по умолчанию показывается имя домена, имя FTP-сервера, его версия и приветственное сообщение. Ничего страшного не видите? А я вижу — хакеру достаточно подключиться на порт 21, чтобы узнать, с каким FTP-сервером он имеет дело.

Дальнейшие действия взломщика легко предсказать. Я бы на его месте поискал во всех базах уязвимостей информацию о дырах в этой версии wu-ftpd. А администратору остается молиться, чтобы я не нашел нужных эксплоитов или чтобы найденная дыра оказалась незначительной и не позволила бы мне ничего сделать с его системой.

После прочтения строки приветствия можно посылать команды FTP-серверу. Но ничего серьезного выполнить не удастся, пока вы ему не представитесь. Для этого нужно выполнить команды FTP: `USER` с параметром `имя_пользователя`, а затем `PASS`, указав пароль.

Серверы FTP позволяют работать с тремя типами авторизации: действительной, гостевой и анонимной. В первом случае вы должны передать серверу реальное имя и пароль пользователя, которому разрешен доступ к серверу. Тогда после выполнения команды `USER` вы увидите сообщение о необходимости ввести правильный пароль для указанного пользователя:

```
331 Password required for flenov
```

В случае с анонимным доступом в качестве имени нужно указать `Anonymous` (команда `USER Anonymous`). В ответ на это сервер вернет нам сообщение:

```
331 Anonymous access allowed, send identity (e-mail name) as password.
```

Здесь говорится, что анонимный доступ разрешен, и вы должны передать в качестве пароля адрес E-mail. Честно говоря, можно ввести и чужой E-mail, например своего соседа. Это сервер проконтролировать не сможет. Некоторые серверы вообще не проверяют корректность адреса даже по простым шаблонам, и им можно передавать что угодно.

На практике анонимный доступ обладает минимальными возможностями чтения файлов и каталогов и используется только в открытых файловых архивах. Имя `Anonymous` чаще всего используют при публикации через FTP открытых документов для всеобщего доступа. Например, разработчики программ создают анонимные FTP-серверы, чтобы пользователи могли скачать с них последние версии программ или обновления к ним.

Реальный (действительный) пользователь может путешествовать по всей файловой системе, и его возможности ограничены только правами доступа учетной записи, по которой он авторизовался на сервере.

Гостевой доступ по своим правам является чем-то промежуточным между анонимным и действительным. По сравнению с анонимным пользователем гость обладает большими правами, и ему может выдаваться разрешение на загрузку файлов на сервер, но, в отличие от действительного, он работает только в своем каталоге. Например, если гостю назначен каталог `/home/guest`, то он сможет работать с файлами и подкаталогами этого каталога, но выше него подняться не сможет. В качестве гостевого вы можете определить любое имя.

Обратите внимание, что пароль в команде `PASS` передается в абсолютно открытом виде. Это серьезная проблема. В каждой главе, где мы рассматриваем какой-либо сервис, доводится сталкиваться с открытой передачей данных. Ну что поделаешь, на заре рождения Интернета никто не думал о хакерах. Теперь приходится выдумывать разные методы, чтобы спрятать пароль.

Если ваш сервер обслуживает только анонимные соединения, то пароли могут передаваться как угодно. При такой аутентификации любой пользователь и так может подключиться к серверу, указав любой адрес E-mail в качестве пароля. Но подобные серверы организуются только для работы с общедоступными ресурсами/файлами. При наличии важной информации доступ осуществляется по паролю, и необходимо сделать так, чтобы он шифровался. Для этого можно использовать `stunnel` или уже готовый протокол SFTP, о котором мы говорили в *разд. 5.3.7*.

Интересное решение я видел на одном из публичных веб-серверов еще свыше 10 лет назад. Для того чтобы загрузить данные на сервер, необходимо было зарегистрироваться, заполнив веб-форму с личными данными. Зарегистрировавшемуся посетителю выдавался пароль на доступ, который действовал только в течение одной сессии. После этого пароль уничтожался, и повторное подключение становилось невозможным. Файлы закидывались в специальный каталог, куда была разрешена только запись. Самим файлам устанавливались права только для чтения и записи, а выполнение оставалось недоступным. В таком случае пароли можно было передавать в открытом виде — даже если хакер узнавал пароль, подключиться он все равно не смог бы.

В современном Интернете загружать файлы все же лучше с помощью HTTP-протокола и веб-форм. Такие варианты отлично работают, особенно если файлы небольшого размера. Доступ по FTP может понадобиться только в крайнем случае или когда размер файла очень большой.

Реализовать одноразовые пароли достаточно просто, если ваш сервер использует подключаемые модули аутентификации PAM (см. *разд. 3.3.3*).

10.1.3. Передача файлов

Поскольку протокол FTP предназначен для работы с разными системами, то для передачи файлов используются два основных режима: текстовый (ASCII) и бинарный.

Допустим, вы хотите переслать текстовый файл с компьютера UNIX на компьютер Windows. В UNIX для текстовых файлов в качестве идентификатора конца строки используется символ <CR> (Carriage Return, возврат каретки, код 13). А в Windows то же самое обозначается двумя символами: тем же <CR> и <LF> (Line Feed, перевод строки, код 10). И если после передачи открыть в Windows текстовый файл, пересланный с компьютера UNIX в бинарном режиме, то из-за отсутствия в нем символа <LF> все строки текста сольются в одну.

На рис. 10.1 показан файл sendmail.cf (используется для конфигурации Sendmail), скачанный с Linux-сервера в бинарном режиме и открытый в программе Windows Notepad. Как видите, очень тяжело разобрать, что здесь и для чего, а вместо перехода на новую строку можно увидеть нечитаемый символ <CR>, представленный в виде пустого квадрата.

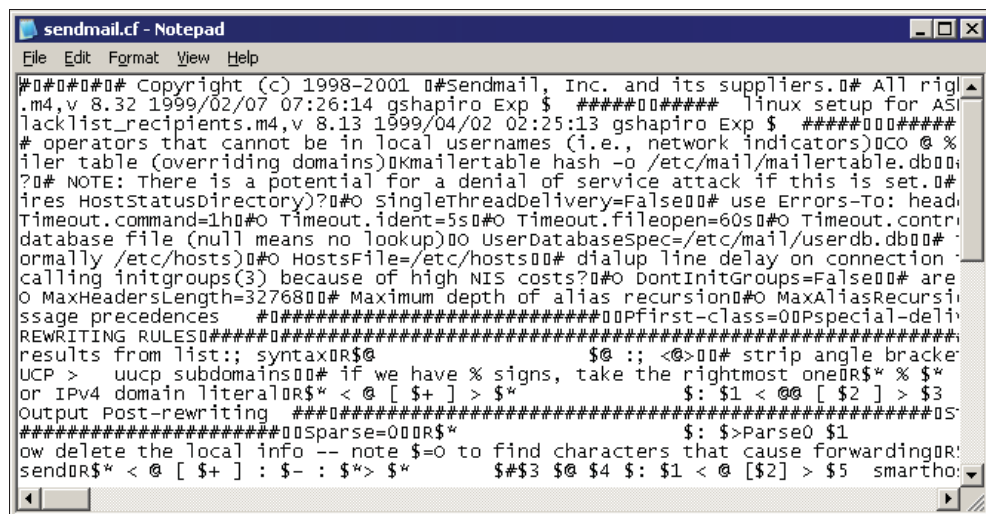


Рис. 10.1. Файл sendmail.cf, полученный с Linux-сервера в бинарном режиме

Чтобы решить проблему конца строки, необходимо скачивать файл с сервера в символьном (ASCII) режиме. В этом случае текст передается построчно, и ОС-получатель сама добавляет нужные символы перевода строки. На рис. 10.2 можно увидеть тот же файл sendmail.cf, полученный с сервера в режиме ASCII. Теперь информация стала читабельной, и все символы перехода на новую строку расставлены самой ОС верно, в соответствии с ее внутренними правилами.

Двоичные файлы (например, картинки или музыку) необходимо передавать в бинарном режиме. Здесь нет различий, в какой ОС создан объект, и он будет правильно воспринят в любой ОС, умеющей работать с этим форматом.

```

# Copyright (c) 1998-2001 Sendmail, Inc. and its suppliers.
# All rights reserved.
# Copyright (c) 1983, 1995 Eric P. Allman. All rights reserved.
# Copyright (c) 1988, 1993
# The Regents of the University of California. All rights reserved.
#
# By using this file, you agree to the terms and conditions set
# forth in the LICENSE file which can be found at the top level of
# the sendmail distribution.
#
#
#####
#####
#####
#####
##### SENDMAIL CONFIGURATION FILE
#####
##### built by leon@pylesos.asp-linux.com.ua on чоп йАО 25 16:59:02 EEST 2002
##### in /home/leon/RPM/BUILD/sendmail-8.11.6/cf/cf
##### using ../ as configuration include directory

```

Рис. 10.2. Файл sendmail.cf, полученный с Linux-сервера в ASCII-режиме

Ну а если передать из Linux в Windows в текстовом режиме двоичный файл, то любой символ <CR> (а он может встретиться и в таком файле) будет заменен на пару символов <CR> и <LF>, и после этого файл может стать нечитаемым.

10.1.4. Режим канала данных

Мы уже говорили о том, что для работы с FTP-сервером необходимы два канала. Порт 21 является управляющим, и по нему передаются только команды FTP. Для передачи файлов создается отдельное соединение. Этот процесс можно описать следующим образом:

1. Программа-клиент открывает порт на компьютере, куда сервер должен передать содержимое файла.
2. Серверу направляется запрос на скачивание файла и сообщается порт и IP-адрес клиентского компьютера, с которым он должен соединиться.
3. Сервер инициализирует соединение с компьютером клиента и начинает передачу данных.

Такой режим называется *активным*, потому что соединение устанавливает сервер. Проблема кроется в последнем действии. Если у вас установлен сетевой экран, то, скорее всего, любые подключения извне запрещены, чтобы хакер не смог подобраться к компьютерам вашей сети. Соединения должны инициализировать только ваши компьютеры, а не внешние.

В результате в активном режиме через правильно настроенный сетевой экран протокол FTP работать не будет. Если же разрешить внешним программам устанавливать соединения, то придется отключить сетевой экран, и он уже ничего не защитит.

Чтобы решить проблему работы через сетевой экран, был разработан *пассивный* режим. В большинстве серверов и клиентских программ именно его теперь начи-

нают устанавливать по умолчанию, потому что сетевые экраны в наше время встроены уже почти во все ОС.

В пассивном режиме соединение происходит иначе:

1. Клиент запрашивает скачивание или просит принять файл.
2. Сервер осуществляет привязку к порту и сообщает клиенту номер канала, куда необходимо подключиться для получения или отправки данных.
3. Клиент устанавливает соединение с указанным портом, по которому происходит передача данных.

Таким образом, сервер только открывает порт и подготавливается к обмену файлами, а все соединения осуществляются только со стороны клиента. Это уже не противоречит правилам сетевого экрана.

10.2. Сервер ProFTPd

В последнее время все бóльшую популярность завоевывает сервер ProFTPd. Его мы подробно рассматривать не станем, но короткий взгляд бросим. Главный конфигурационный файл этого сервера — `/etc/proftpd.conf`. Кроме того, к его конфигурационным файлам также относится файл `ftprusers`, где описываются пользователи, которым разрешена работа с FTP.

При конфигурировании `proftpd` большинство действий описываются блоками. Например:

```
<Limit SITE_CHMOD>
  DenyAll
  Allow from 127.0.0.1
</Limit>
```

Здесь описывается блок `Limit`, который позволяет задавать ограничение. Что именно ограничивается, указывается сразу после ключевого слова `Limit`. В нашем случае — это операция `SITE_CHMOD`. Внутри блока идут операторы ограничения:

- ☐ `DenyAll` — запрещает доступ со всех компьютеров;
- ☐ `Allow from` — разрешает доступ только с указанного компьютера.

Мы снова соблюдаем правило запрещения всего, что не разрешено явно. Можно также указывать следующие ограничения:

- ☐ `AllowAll` — разрешить доступ со всех компьютеров;
- ☐ `AllowGroup` группы — разрешить доступ с указанных групп;
- ☐ `AllowUser` — разрешить доступ указанным пользователям;
- ☐ `AllowOverwrite` значение — определяет, можно ли перезаписывать уже существующие файлы. В качестве значения можно указывать `on` или `off`.

Для всех пунктов, кроме последнего, есть аналогичные запреты — нужно только поменять `Allow` на `Deny`. Это операторы ограничения. В блоках могут быть и другие

операторы, которых мы рассмотрим чуть позже, а сейчас давайте закончим знакомство с типами блоков.

Чтобы создать сервер с анонимным доступом, нужно описать блок `Anonymous`:

```
<Anonymous путь>
</Anonymous>
```

Анонимным пользователям будет разрешен доступ только к каталогу `путь`, и ничего более.

Чтобы определить доступ к каталогу, можно использовать директиву `Directory`:

```
<Directory путь>
    DenyAll # для примера запретим доступ
</Directory>
```

В этом блоке вы можете указывать директивы, которые относятся к конкретному каталогу. Если нужно указать глобальные параметры, то используйте блок `Global`. Глобальному блоку не нужно указывать каталог — он действует на все.

Еще один блок, который может присутствовать в конфигурации сервера, — `VirtualHost`. В нем указывается адрес сервера.

Теперь рассмотрим параметры, которые можно использовать внутри этих блоков:

- ☐ `DefaultRoot` — корень дерева каталогов FTP, выше которого пользователь не может подняться. По умолчанию пользователь получает доступ к своему домашнему каталогу, а доступ выше запрещен;
- ☐ `DefaultTransferMode` — режим передачи данных. Их всего два: текстовый и бинарный, и в этом параметре можно указать соответственно `ascii` или `binary`;
- ☐ `DisplayConnect имяфайла` — содержимое файла `имяфайла` будет показано в соответствующем сообщении во время соединения клиента с сервером;
- ☐ `DisplayLogin имяфайла` — содержимое файла `имяфайла` будет показано пользователю после успешной регистрации;
- ☐ `MaxClients` — максимальное количество одновременно подключающихся пользователей. Слишком маленькое число позволит хакеру с легкостью реализовать DoS-атаку, исчерпав все возможные соединения;
- ☐ `Order` — порядок, в котором сервер применяет разрешения. Если здесь указано `allow`, `deny`, то разрешающие записи имеют больший приоритет. Если ничего не указано, то доступ разрешается. Если же указано значение `deny`, `allow`, то больший приоритет имеет запрещение;
- ☐ `RootLogin` — разрешение администратору подключаться по FTP. Если здесь указано `on`, то такое действие разрешено, но лучше все же его запретить, указав `off`. FTP — не лучшее решение для администрирования файлов. Если хакер каким-либо образом узнает или подберет пароль, то можете попрощаться со своими данными;
- ☐ `ServerName` — имя сервера, которое будет показываться клиенту;

- `SyslogLevel` — уровень журналирования событий:
 - `emerg` — наиболее важные;
 - `alert` — события;
 - `crit` — критические сообщения;
 - `error` — ошибки;
 - `warn` — предупреждения;
 - `notice` — сообщения;
 - `info` — информация;
 - `debug` — отладочная информация;
- `umask маска_файла маска_каталога` — маска файла для новых файлов и папок, работающая точно так же, как параметр `umask` системы. Маска каталога не является обязательной.

Как видите, уже по параметрам понятно, что к чему, так что ничего сложного здесь нет.

10.3. Еще несколько слов о протоколе FTP

Протокол FTP и серверы различных производителей за время своего существования имели множество проблем с безопасностью. Если сложить проблемы из-за ошибок FTP и программы Sendmail, то результат может затмить даже потери от внедрения вирусов.

Главная проблема протокола FTP заключается в том, что он создавался как дружественный и удобный для пользователя. Вторая проблема — использование двух портов. Авторизация происходит только при подключении на 21-й порт, а работа с каналом для передачи данных происходит без какого-либо подтверждения подлинности клиента.

Если во времена создания FTP-протокола он был действительно необходим для обмена данными, то в настоящее время от него следует избавляться. Если вы хотите дать пользователям возможность только скачивать информацию, то обратите внимание на протокол HTTP. С его помощью можно даже организовать загрузку файлов на сервер.

Если необходим обмен данными в локальной сети, то можно использовать Samba-сервер или, опять же, HTTP. Многие администраторы не хотят настраивать веб-сервер только ради обмена данными и устанавливать на него потенциально опасные сценарии. Но нельзя забывать, что FTP также может оказать медвежью услугу. Из двух зол нужно выбирать меньшее. Если у вас уже работает веб-сервер, то максимально используйте его возможности, и тогда можно будет закрыть 21-й порт, тем самым оградив себя от вероятных неприятностей, которые могут с его участием произойти.

Если вам лично необходим доступ к серверу для работы с файлами удаленно, то советую использовать пакет SSH и встроенный протокол SFTP, который шифрует данные, поскольку такое соединение перехватить намного сложнее.



ГЛАВА 11

DNS-сервер

Каждый компьютер, подключенный к сети, должен иметь свой адрес, чтобы его можно было найти и обмениваться с ним данными. Это похоже на телефонные сети. Чтобы кому-нибудь позвонить, нужно знать номер его телефона, иначе коммутатор не сможет понять, с кем вас соединить.

Когда вы хотите получить веб-страничку с сервера, то необходимо знать его адрес. В запросе нужно указать свои координаты, чтобы сервер знал, куда вернуть требуемую страничку. Здесь просматривается аналогия с почтой. Отправляя письмо, вы указываете адрес, а если хотите, чтобы вам ответили, то должны указать и обратный адрес.

В эпоху зарождения сетей компьютеров в них было мало, поэтому для адресации был выбран самый простой вариант — числа. Я думаю, что если бы Интернет появился только сейчас, то приняли бы точно такое же решение. Но с увеличением количества компьютеров сетевая общественность стала осознавать, что помнить больше 20 адресов по номерам невозможно, поэтому было принято решение найти какой-либо способ, упрощающий запоминание.

Изменять принцип нумерации было поздно, да и выбранная система IP-адресов оказалась очень удобна для обработки на программном уровне компьютерами и сетевыми устройствами. Немного поколдовав, умные люди придумали создать централизованную базу данных, устанавливающую соответствие IP-адресов и символьных имен. Таким образом, пользователь оперирует именами узлов, а программа находит нужный IP-адрес в этой базе и уже по нему соединяется с другим компьютером или сервером.

Этот метод очень сильно упростил задачу. Раньше, чтобы найти сервер, нужно было четко знать его IP-адрес. Теперь в простой ситуации можно обойтись даже без поисковых систем типа Yahoo или Google. Например, если нужен сайт корпорации Microsoft, то можно попробовать набрать в браузере адрес **www.microsoft.com**. Таким путем веб-серверы фирм чаще всего можно найти просто по их имени, а не запоминать лишнюю информацию.

ся запрос на поиск домена с именем **flenov**. Если такой найден, то мы получим адрес DNS-сервера, который обслуживает домен **flenov.info**. И уже ему направляется запрос на получение адреса домена **www.flenov.info**. Результатом будет IP-адрес сервера, которому присвоено имя **www.flenov.info**.

Все эти действия происходят прозрачно для конечного пользователя, и когда вы набираете в браузере адрес, то никогда не увидите этих тонкостей. О том, что идет поиск IP-адреса по имени, можно узнать только по подсказке, которая высвечивается в строке состояния обозревателя.

В сети есть множество кэширующих серверов, выполняющих свои функции автоматически. Достаточно только установить такой сервер, сделать основные настройки и запустить. Кэширующие серверы обмениваются между собой информацией и позволяют найти любой адрес на ближайшем узле, не обращаясь к основной базе данных. Например, у вашего интернет-провайдера чаще всего есть свой DNS-сервер. Когда вы обращаетесь на какой-нибудь символьный адрес, то запрос сначала идет на сервер провайдера, затем по цепочке передается другому серверу, и так до тех пор, пока нужная запись с IP-адресом не будет найдена, если, конечно же, имя указано верно. При такой организации запросов адрес запрашиваемого имени может быть получен от ближайшего DNS-сервера, в кэше которого сохранилась необходимая информация.

Серверы DNS могут и, наоборот, по IP-адресу сказать нам его символьный адрес. В этом случае IP-адрес тоже переворачивается. Например, если нужно узнать имя сервера 190.1.15.77, то в запросе к DNS будет виден адрес 77.15.1.190 и добавлен суффикс `in-addr.arpa`: **77.15.1.190.in-addr.arpa**.

11.2. Локальный файл `hosts`

Мы уже знаем, что изначально для сопоставления имен и адресов использовался файл `/etc/hosts`. Это текстовый файл с записями типа:

```
127.0.0.1 localhost.localdomain localhost
192.168.77.1 FlenovM
```

Каждая строка — это соответствие IP-адреса его имени. По умолчанию в файле будет всего две строки, как в приведенном примере. Первая строка — это «петля». Напоминаю, что во всех компьютерах имя `localhost` и IP-адрес 127.0.0.1 указывают на текущую машину. Это значит, что, если нужно выполнить `ping`, адресовав запросы на локальный компьютер, можно написать:

```
ping 127.0.0.1
```

Во второй записи устанавливается соответствие между IP-адресом, заданным для вашего сетевого интерфейса, и символьным именем. Здесь моей сетевой карте присвоен адрес 192.168.77.1, и ему соответствует имя `FlenovM`. Это значит, что при выполнении команды `ping` можно указывать или IP-адрес, или имя компьютера.

Следующие две команды идентичны:

```
ping 192.168.77.1  
ping FlenovM
```

При выполнении второй команды сначала происходит обращение к файлу `/etc/hosts`, который вернет программе адрес `192.168.77.1` и уже на него направится эхо-запрос.

А что будет использоваться для поиска адреса первым: файл `/etc/hosts` или DNS-база данных? Это зависит от настроек ОС. Посмотрим на файл `/etc/host.conf`. В нем находится строка:

```
order hosts,bind
```

Директива `order` как раз и задает порядок просмотра. В нашем случае на первом месте находится файл `/etc/hosts`, и только после этого будет запущена команда `bind` для выполнения запроса к DNS-серверу. Что это нам дает? А то, что можно увеличить скорость доступа к основным серверам. Допустим, что вы каждый день посещаете сайт **www.redhat.com**. При этом каждый раз происходит запрос к DNS-серверу, что может приводить к задержке в пару секунд перед началом загрузки страницы. Чтобы ускорить этот процесс, можно вручную прописать в файле `/etc/hosts` следующую запись:

```
209.132.183.105          www.redhat.com
```

ВНИМАНИЕ!

Адрес `209.132.183.105` действительно соответствует сайту **www.redhat.com** на момент подготовки этого издания книги, но может измениться.

Если сайт по каким-либо причинам перестал загружаться, то необходимо удалить соответствующую запись из файла `hosts` и с помощью команды `ping redhat.com` проверить связь с сервером, а заодно и уточнить его адрес. В ответ на эту директиву на экране обязательно отображается реальный IP-адрес, с которым происходит обмен эхо-сообщениями. Благо IP-адреса у большинства сайтов изменяются редко, так что, один раз добавив такую запись в локальный файл `/etc/hosts`, можно сэкономить достаточно много времени и нервов в случае проблем с DNS-сервером, потому что запросов к нему отправляться не будет.

11.3. Внешние DNS-серверы

Если в локальном файле `/etc/hosts` не найдено записи о нужном имени, то компьютер должен запросить эту информацию у DNS-сервера. Для этого нужно знать IP-адрес этого самого сервера. Как система его узнает? Из файла `/etc/resolv.conf`, который должен выглядеть примерно следующим образом:

```
search FlenovM  
domain domain.name  
nameserver 10.1.1.1  
nameserver 10.1.1.2
```

В первой строке находится команда `search` с параметром (сервер поиска имени хоста). В вашем файле, скорее всего, есть эта запись, и в качестве сервера вписано имя вашего компьютера. Здесь может быть перечислено несколько серверов, разделенных пробелами или символами табуляции. Например:

```
search FlenovM MyServer
```

Поиск на локальном сервере происходит довольно быстро, а вот на удаленных — может отнять достаточно много времени.

Вторая строка содержит команду `domain` с параметром. Пользователи иногда любят задавать имя компьютера без указания домена — например, `redhat` вместо `redhat.com`. Вы должны задать полное имя узла. Чаще всего этот параметр настраивается в локальных сетях со специфичным именем домена.

Оставшиеся две строки содержат команду `nameserv` с параметром. Это внешний DNS-сервер, которому будут направляться запросы. В системе их может быть несколько (в настоящее время для большинства дистрибутивов не более трех). Они будут опрашиваться в порядке указания в файле, пока искомый адрес не будет найден.

В большинстве случаев достаточно и одного сервера, потому что все они работают рекурсивно. Но я рекомендую указывать два. Бывают случаи, когда один DNS-сервер выходит из строя, тогда второй спасает положение и вступает в работу.

11.4. Настройка DNS-сервиса

В настоящее время наиболее распространенным сервисом DNS для Linux является `bind`. Для этого сервиса существует программа `bindconf`, которая имеет графический интерфейс и проста в использовании. Зайдите в графическую оболочку и в консоли выполните команду:

```
bindconf &
```

Знак `&` говорит о том, что, запустив программу, консоль не должна дожидаться ее завершения. Это очень удобно при запуске графических утилит, чтобы они не останавливали работу консоли. Учтите, что если закрыть окно консоли, то все программы, запущенные с ключом `&`, тоже закроются.

На рис. 11.2 изображено запущенное приложение для настройки DNS-сервиса. В центре показано окно, которое открывается по нажатию кнопки **Добавить**. Как видите, достаточно только выбрать тип зоны и ввести имя домена — и все готово.

Несмотря на наличие простой графической программы, мы рассмотрим конфигурационные файлы, которые могут использоваться сервисом DNS. Их прямое редактирование позволит сделать более тонкую настройку, и вы лучше будете понимать процесс работы DNS.

Настройка DNS-сервера начинается с файла `/etc/named.conf`. Пример его содержимого приведен в листинге 11.1.

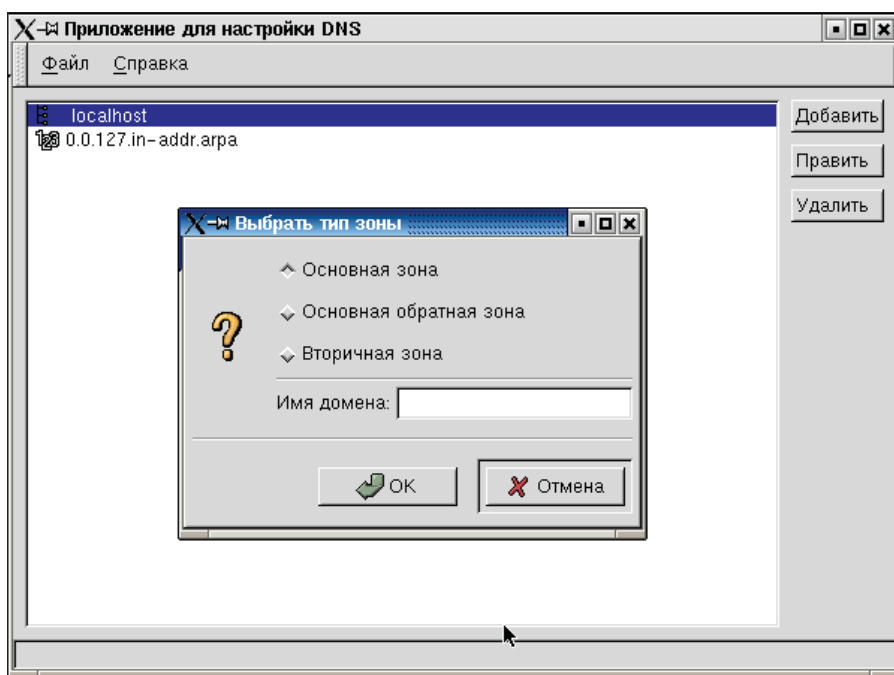


Рис. 11.2. Приложение для настройки сервиса DNS

Листинг 11.1. Пример содержимого файла /etc/named.conf

```
options {
    directory "/var/named/";
};

zone "." {
    type hint;
    file "named.ca";
};

zone "sitename.com" {
    type master;
    file "sitename.zone";
};

zone "10.12.190.in-addr.arpa" {
    type master;
    file "10.12.190.in-addr.arpa.zone";
};
```

В этом примере файл разбит на четыре раздела, каждый из которых имеет следующий формат:

```
тип имя {  
    Параметр1;  
    Параметр2;  
    ...  
};
```

Давайте рассмотрим назначение разделов. Первым идет `options`:

```
options {  
    directory "/var/named/";  
};
```

В фигурных скобках только один параметр — `directory`, который указывает на домашний каталог DNS-сервера. Все его файлы будут располагаться там.

Остальные разделы имеют тип `zone` и через пробел в кавычках содержат имя зоны. В каждом из них по два параметра: `type` (определяет тип зоны) и `file` (файл, в котором содержится описание).

Самая первая зона в нашем примере — `zone "."`. Что это за зона в виде точки? Вспомните структуру DNS, которую мы рассматривали в начале главы. В базе данных DNS так обозначается корень иерархии. Получается, что раздел описывает корневую зону. Тип зоны `hint`, т. е. здесь хранятся лишь ссылки на DNS-серверы. Поскольку это корневая зона, то и ссылки будут на корневые серверы.

В параметре `file` указывается имя файла, содержащего все ссылки на корневые серверы. В системе этого файла может и не быть, потому что информация в нем подвержена изменениям, и лучше всего получить его последнюю версию с сервера **internic.net**. Для этого выполните команду:

```
dig @rs.internic.net . ns > named.ca
```

А можно скачать этот файл непосредственно с FTP-сервера **internic.net** по адресу **ftp://ftp.rs.internic.net/domain/named.root** и поместить в каталог `/var/named`.

Перейдем к следующему разделу `zone "sitename.com"`. Здесь описывается зона **sitename.com**. Тип записи `master`, значит, ваш DNS-сервер будет главным, а все остальные станут только сверяться с ним и кэшировать информацию. Сведения об этой зоне будут храниться в файле `sitename.zone` рабочего каталога. В нашем случае это: `/var/named`.

Следующая зона описывает обратное преобразование IP-адресов `190.12.10.*` в имена. Тип записи — снова `master`, и в последней строке указан файл, где будет находиться описание зоны.

11.5. Файлы описания зон

Теперь посмотрим, что у нас находится в каталоге `/var/named`. Судя по файлу конфигурации `/etc/named.conf`, у нас здесь должно быть три файла:

- `named.ca` — хранит ссылки на корневые серверы. Этот файл просто забирается с сервера **internic.net**, поэтому его редактировать не стоит, и даже не будем на нем останавливаться;


```
IN      NS      localhost.

1       PTR     servername.com.
2       PTR     mail.servername.com.
```

Большая часть этого файла нам уже знакома. Самое интересное хранится в последних двух строках. Здесь находится связка IP-адресов и имен серверов. Не забываем, что файл отвечает за сеть с адресами 190.12.10.*. Звездочка заменяется на число, стоящее в первой колонке, а имя, соответствующее этому адресу, указано в последнем столбце. По этому файлу мы видим следующие соотношения:

190.12.10.1 = **servername.com**.

190.12.10.2 = **mail.servername.com**.

Еще раз напоминаю, что точка в конце символьного адреса обязательна.

Для получения дополнительной информации по DNS рекомендую прочитать документы RFC 1035, RFC 1712, RFC 1706.

11.7. Безопасность DNS

Если посмотреть на задачи, которые решает служба, то кажется, что ничего подозрительного в ней нет и хакер не сможет с ней ничего сделать. Как бы не так. Были случаи, когда DNS-серверы выводили из строя. Тогда обращение по именам становилось невозможным, а значит, сетевые программы переставали работать. Пользователи не привыкли использовать IP-адреса, поэтому падение DNS для них смертельно.

Помимо вывода из строя сервера, хакер может узнать многое о структуре сети, используя DNS. Чтобы этого не произошло, желательно использовать два DNS-сервера:

- ☐ общедоступный, содержащий строки, необходимые для работы удаленных пользователей с общими ресурсами;
- ☐ локальный, видимый только пользователям вашей сети и содержащий все необходимые для их работы записи.

На локальном сервере можно так настроить сетевой экран, чтобы он воспринимал лишь локальный трафик и игнорировал любые попытки обращения из Всемирной сети. В этом случае злоумышленнику будет проблематично не только посмотреть базу данных DNS, но и нарушить работу сервера. Таким образом, все локальные пользователи будут лучше защищены от нарушения работы DNS и смогут спать спокойным сном под охраной сетевого экрана. Хотя спать на рабочем месте и нехорошо.

Для каждого первичного можно завести по одному вторичному серверу. Это позволит распределить нагрузку между ними и уменьшить время отклика и, конечно же, повысить отказоустойчивость. При выходе из строя одного из серверов второй возьмет на себя его функции и не позволит сети остаться без удобной возможности адресации к компьютерам по имени.

Использование парных серверов позволяет повысить производительность и безопасность. Сервисы DNS под Linux не требовательны к оборудованию. В моей сети работают четыре сервера на базе Red Hat Linux в текстовом режиме на компьютерах Pentium с частотой от 400 до 700 МГц. Когда-то это были офисные машины, но их мощности перестало хватать, и я превратил старое «железо» в DNS-серверы. Для выполнения этой задачи столь древней техники более чем достаточно и хватит на ближайшие годы. Таким образом, старому компьютеру можно дать новую жизнь, причем достаточно долгую, а, главное, для компании такое решение окажется приемлемым по цене.

Однако технология создания вторичных серверов опасна. С помощью утилиты `host` хакер может добыть полную информацию о содержимом базы данных основного сервера, как это делают вторичные серверы для обновления своей базы.

Для получения базы взломщик может выполнить следующую команду:

```
host -l server.com ns1.server.com
```

В ответ на такой запрос хакер получит из базы данных все записи о сервере **server.com**. Чтобы предотвратить это, необходимо явно прописать адреса вторичных серверов в файле `named.conf`. Для этого в разделе `options{...}` добавляем строку:

```
allow-transfer {192.168.1.1;}
```

Такого рода ограничение можно поместить и в описание отдельных зон, но лучше сделать это один раз в глобальных опциях. Если у вас нет вторичных серверов, то запретите перенос данных в эту зону следующей строкой:

```
allow-transfer {none;}
```

Серверы DNS могут быть подвержены атаке DoS. В ноябре 2002 года был произведен один из самых громких налетов на Интернет такого типа. Атака шла сразу на несколько корневых серверов. Если бы работу DNS выполнял только один сервер, то через некоторое время после начала штурма Интернет стал бы недоступным. Сеть осталась работоспособной благодаря следующим факторам:

- ☐ избыточности серверов, которые дублируют записи;
- ☐ наличию кэширующих серверов;
- ☐ установке прокси-серверов, которые также умеют кэшировать записи DNS.

В остальном защита DNS идентична защите любого другого сервиса и ОС Linux. Как я уже говорил, лучший сервер — это тот, который выполняет узкую задачу. В нем меньше открытых портов и запущенных сервисов, поэтому его труднее взломать. Единственная сложность — большое количество серверов усложняет процесс обновления ОС. В Linux тоже бывают ошибки, и их надо исправлять. Под эту процедуру должны попасть все компьютеры, в том числе и серверы DNS.



ГЛАВА 12

Мониторинг системы

Первоначальная задача администратора — установить систему, правильно распределить права доступа и настроить все необходимые сервисы. После этого многие из нас складывают ручки и начинают гонять монстров по коридорам виртуального мира Doom 3 (или что там сейчас за окном популярно). Если быть таким администратором, то рано или поздно сервер будет взломан, — любая система требует наблюдения. И сервер не исключение — он как ребенок: любит, чтобы за ним наблюдали, иначе нашалит.

Чтобы уменьшить вероятность проникновения в систему постороннего и обезопасить себя от недобросовестных пользователей сети, нужно постоянно держать сервер под контролем. Большинство взломов происходят из-за того, что администраторы не удосужились обновить какие-либо сервисы и установить заплатки. Если взломщики узнают об уязвимости раньше администраторов, то начнут ломать все сервисы с этой ошибкой, что попадают на глаза.

Хороший администратор может и должен своевременно найти информацию об уязвимости и сделать все необходимое для предотвращения любой атаки. В этих целях следует производить регулярный мониторинг системы и поиск «узких мест». Иногда хакеры проникают в систему и долгое время находятся в ней, не проявляя видимой активности.

Если взлом произошел, то тут уже задача не просто восстановить работу, а предотвратить повторное вмешательство. Компании Sony, Apple и другие крупные бренды, будучи подвергнуты взломам, отключали сервисы для полного исследования и восстановления. Я же видел людей, которые после взлома просто восстанавливают удаленные файлы и продолжают заниматься своими делами в надежде, что такое больше не повторится. Это ошибка, потому что хакер уже почувствовал себя безнаказанным и обязательно вернется.

К следующему его приходу вы должны быть готовы. Сделайте все возможное, чтобы найти сведения о взломщике и о том, как он проникает на сервер, и постарайтесь блокировать повторную атаку. Также просмотрите все последние бюллетени ошибок (BugTraq) в поисках информации о «дырах» в вашей версии ОС и в установленных сервисах.

Не будем дожидаться ситуации, когда злоумышленник проникнет в систему и мы потеряем сон. Давайте рассмотрим действия, которые можно произвести еще до взлома, пока система работает нормально. Это поможет повысить ее безопасность.

Все, о чем мы будем говорить в этой главе, необходимо делать и до проникновения в систему, и после. Взломщики иногда оставляют «потайные двери» (например, устанавливают SUID-бит на программу, для которой он не должен быть установлен), поэтому желательно регулярно сканировать систему на предмет безопасности описанными далее методами. Особенно это касается новой системы (сразу после инсталляции и настройки), но такие проверки следует производить и после обновления, добавления программ или сразу после взлома.

У меня на серверах каждую ночь осуществляется тестирование сторонней компанией всех возможных взломов. Это требование клиента, и оно очень даже правильное. Сторонняя компания каждую ночь запускает автоматическое тестирование, и оно не должно приводить к сбоям или ошибкам в работе серверов и сайтов, которые я обслуживаю.

Один раз я слышал, что такой подход может привести к паранойе. Согласен, требования безопасности, которые я привожу, достаточно жесткие, но уж лучше 10 раз все проверить, перепроверить и установить один лишний замочек, чем потом разгребать мусор после взлома.

Существует много различных программ мониторинга, и описывать их все совершенно бесполезно и невозможно. Я расскажу только о нескольких, возможно не самых мощных, и постараюсь немного уделить внимание функциям, встроенным в ОС.

12.1. Автоматизированная проверка безопасности

Практически каждый день специалисты по безопасности находят в разных системах недочеты и, откровенно говоря, «дыры» или даже «пробоины». Весь этот перечень выкладывается в отчетах BugTraq на разных серверах. Я уже советовал посещать www.securityfocus.com, чтобы следить за новостями, и сейчас не отказываюсь от своих слов. Новинки действительно надо искать на подобных серверах, но ведь есть еще ворох старых уязвимостей, которые могли остаться на сервере незалатанными. Как же поступить с ними? Неужели придется качать все эксплойты и «руками» проверять каждую «дыру»? Просто старайтесь, чтобы на ваших серверах стояли самые последние версии ПО, — в последних его версиях старые ошибки должны быть устранены, так что тут нужно просто положиться на разработчиков.

Есть еще такой класс уязвимостей, который связан с неверной конфигурацией. ОС и ее сервисы могут быть безопасны в поставке по умолчанию, но если что-то сконфигурировать неверно, ОС может это пропустить. Как же проверять подобные вещи? Существует громадное количество программ для автоматизации тестирования

Так и при автоматическом сканировании: один сканер может позвонить по телефону, а другой — постучит в дверь. Они оба хороши, но в конкретных случаях при разных конфигурациях сканируемой машины могут быть получены разные результаты.

Существуют три метода автоматического определения уязвимости: сканирование, зондирование и имитация. В первом случае сканер собирает информацию о сервере, проверяет порты, чтобы узнать, какие установлены сервисы/демоны, и на их основе выдает отчет о потенциальных ошибках. Например, сканер может проверить сервер и увидеть на порту 21 работающую службу FTP. По строке приглашения (если она не была изменена), выдаваемой сервером при попытке подключения, можно определить его версию, которая сравнивается с базой данных. И если в базе присутствует уязвимость для этого сервера, то пользователю выдается соответствующее сообщение.

Сканирование — далеко не самый точный процесс, потому что автоматическое определение легко обмануть. Некоторые погрешности в сервисах проявляются только при определенных настройках, и если ваши настройки отличаются от предусмотренных сканером, ошибка может и не обнаружиться. Кроме того, при установке, например, нового FTP-сервера, который еще не известен сканерам, он будет опробован лишь на уже известные ошибки других серверов.

При *зондировании* сканер не обследует порты, а проверяет программы на наличие в них уязвимого кода. Этот процесс похож на работу антивируса, который просматривает программы в поисках вирусных фрагментов. Ситуации похожи, но искомые объекты различны. Метод хорош, но одна и та же ошибка может встречаться в нескольких программах. И если код в них разный, то сканер ее не обнаружит. Очень часто программисты разных фирм допускают в своих продуктах одни и те же промахи, а методом зондирования анализатор может не найти подобную уязвимость только потому, что для новой версии продукта нет записей в базе данных.

В процессе *имитации* программа моделирует атаки из своей базы данных. Например, в FTP-сервере может возникнуть переполнение буфера при реализации определенной команды. Сканер не будет выявлять версию сервера, а попытается выполнить инструкцию. Конечно же, выявление ошибки приведет к зависанию, но вы точно будете знать о ее наличии или отсутствии. Имитация — самый долгий, но и самый надежный способ, потому что если программе удалось взломать какой-либо сервис, то и у хакера это получится.

Когда вы проверяете систему, обязательно отключайте сетевые экраны. Если блокирован доступ, то сканер не протестирует нужный сервис. В этом случае анализатор сообщит, что ошибок нет, но реально они могут быть. Конечно же, это не критичные ошибки, если они под защитой сетевого экрана, но если хакер найдет потайной ход и обойдет брандмауэр, то уязвимость станет опасной.

Отключать сетевой экран совсем и для всех не надо — можно просто разрешить все действия, приходящие с IP-адреса машины, которая производит тестирование.

Дайте сканеру все необходимые права на доступ к тестируемой системе. Например, некоторые считают, что наиболее эффективно удаленное сканирование, когда атака

с SUID- и SGID-битом, выполняющиеся от имени других пользователей, но таких меньшинство.

Если вы видите, что программа вами не используется, то ее стоит удалить или снять с нее опасные биты. Если, по вашему мнению, каким-то программам эти биты все же нужны, подумайте еще раз. Может, все-таки стоит от чего-то отказаться? Например, программа `ping` не является обязательной для сервера, и у нее бит SUID можно снять.

Если и после корректировки привилегированных программ осталось все равно много, то советую для начала убрать эти биты со всех программ. Конечно же, тогда пользователи не смогут монтировать устройства. Но нужно ли это им? И если уж возникнет острая необходимость, то всегда можно вернуть им такую возможность, восстановив SUID-бит.

Почему необходимо регулярно проверять файлы на наличие SUID- и SGID-бита? Взломщики, проникая в систему, очень часто стремятся укрепиться в ней, спрятавшись и при этом иметь максимальные права. Для этого может использоваться простейший метод — установка SUID-бита на интерпретатор команд `bash`. В этом случае интерпретатор для любого пользователя будет выполнять команды с правами `root`, и взломщику для выполнения любых действий достаточно находиться в системе с гостевыми правами.

Следите за любыми появляющимися новыми программами с битами SUID или SGID и убирайте все, что вызывает подозрение.

Использование битов SUID или SGID — очень старый прием взломщиков, и я сейчас редко о нем слышу, но нет гарантии, что хорошо забытая проблема не станет новой.

12.3. Проверка конфигурации

Мы рассмотрели весьма много правил конфигурирования системы, однако помнить их все невозможно. Настройка системы — сложный процесс, во время которого очень легко ошибиться. Но так как есть определенные правила конфигурирования, то существует и возможность автоматизировать проверку.

Как мы уже знаем, имеются программы, которые могут проверять конфигурацию. Это должно происходить на компьютере локально. К настоящему времени написано много утилит для автоматизации контроля. Некоторые из них устарели и давно не обновлялись, а какие-то появились недавно и еще только наращивают свою функциональность (количество производимых проверок).

Я решил познакомить вас с программой `lsat`. Ее история не слишком длинна, но за счет частого обновления на начальном этапе (последнее обновление вышло в 2014 году) возможности программы серьезно выросли, а благодаря модульной архитектуре расширение функций происходит легко и быстро.

Программа `lsat` поставляется в исходных кодах, и найти ее можно по адресу: usat.sourceforge.net. На сайте доступны TGZ- и ZIP-версии архивов программы.

Пока слова о безопасности отсутствуют, есть только информация о том, что сканировалось. Все самое интересное после тестирования можно найти в файле `./lsat-0.9.2.tgz/lsat.out`. Я прогнал программу `lsat` на своей системе сразу после установки и получил документ размером 190 Кбайт.

В этом выходном файле вы найдете множество советов. Так, в самом начале можно увидеть рекомендации по пакетам, которые нужно удалить:

```
*****
```

```
Please consider removing these packages.
```

```
sendmail-8.11.6-15.asp
```

```
portmap-4.0-41
```

```
bind-utils-9.2.1-1.asp
```

```
nfs-utils-0.3.3-5
```

```
pidentd-3.0.14-5
```

```
sendmail-devel-8.11.6-15.asp
```

```
sendmail-cf-8.11.6-15.asp
```

```
ypbind-1.10-7
```

```
ypbind-1.10-7
```

В самом деле, некоторые пакеты можно отнести к разряду не очень надежных. Например, в программе `sendmail` регулярно находят ошибки, поэтому `lsat` и предлагает ее «вычистить».

Мне также очень понравилась в выходном файле следующая запись:

Разработчик программы посчитал, что загрузка в графическом режиме хуже для безопасности. Действительно, это лишние программы и дополнительные проблемы. Текстовый режим не требует столько ресурсов, работает меньше программ, а значит, он быстрее и безопаснее.

Если опуститься по файлу немного ниже, то вы увидите список всех файлов в системе с установленными битами `SUID` и `SGID`. Так что при использовании программы `lsat` нет смысла самостоятельно заниматься поисками потенциально опасных программ.

Еще немного ниже идет список общедоступных файлов:

```
*****
```

```
This is a list of world writable files
```

```
/var/lib/texmf/ls-R
```

```
/var/www/html/cache/archive/index.html
```

```
/var/www/html/cache/categories/category.cgi
```

```
/var/www/html/cache/categories/index.html
```

```
/var/www/html/cache/download/download-2-1.cgi
```

```
/var/www/html/cache/download/download-3-1.cgi
```

```
/var/www/html/cache/download/download-4-1.cgi
```

Это те файлы, изменять которые имеют право любые пользователи системы, даже с минимальными правами. В идеале таких записей вообще не должно быть. В любой файл должны иметь право писать только их владельцы или, в крайнем случае, пользователи группы, но никак не все остальные.

А это еще одна причина, по которой администратор не должен входить в систему под учетной записью `root`. Лучше всего работать как простой пользователь, а когда не хватает прав, то переключаться на привилегированного. На такой случай я создал учетную запись, для которой установил `UID` равным нулю. Она позволяет получить доступ ко всей системе, и при этом имеет имя, отличное от `root`, и не вызывает подозрений, когда я под ним работаю. Так что в моей системе никогда нельзя увидеть пользователя `root`.

users

Команда `users` также позволяет вытащить из журнала `/var/run/utmp` список всех пользователей, которые сейчас зарегистрированы в системе.

В этом журнале информация хранится временно — только в период присутствия пользователя. Когда он выходит из системы, соответствующая запись удаляется. После этого выяснить, кто и когда работал, удастся только из журнала `/var/log/wtmp`. Это также бинарный файл, и его содержимое можно увидеть с помощью специализированных программ.

last

Команда `last` позволяет выяснить, когда и сколько времени определенный пользователь находился в системе. В качестве параметра ей передается интересующее имя. Например, следующая директива отображает время входа и продолжительность нахождения в системе пользователя `robert`:

```
last robert
```

Выполнив команду, вы увидите на экране примерно следующий текст:

```
robert      tty1      Thu Dec 2 12:17 — 12:50 (00:33)
```

По этой записи можно понять, что `robert` находился за терминалом (`tty1`), зашел в систему 2 декабря на 33 минуты (с 12:17 до 12:50). Если пользователь работал не локально, а через сеть, то будет отображена информация о хосте, с которого он входил в систему.

Если выполнить эту команду для себя, то может вывалиться такой список, что читать его будет невозможно, потому что вы довольно часто работаете в системе. Чтобы ограничить выводимые данные, можно указать ключ `-n` и количество отображаемых строк. Например, следующая команда выдаст информацию о последних пяти входах:

```
last -n 5 robert
```

history

Команда `history` выводит историю выполненных в консоли команд. Список может быть достаточно большим. Вы можете ограничить список, указав после имени

Список состоит из четырех колонок:

- ☐ имя пользователя из файла `/etc/passwd`;
- ☐ порт или терминал, на который происходило подключение;
- ☐ адрес компьютера, если вход был по сети;
- ☐ время входа.

С помощью `lastlog` удобно контролировать системные записи. У них дата последнего входа должна быть `**Never logged in**`, потому что под ними нельзя войти в систему (в качестве командных оболочек установлены `/bin/false`, `/dev/null`, `/sbin/nologin` и др.). Если вы заметили, что кто-либо вошел в систему через одну из этих учетных записей, то это значит, что хакер использует ее, изменив настройки.

Простая замена командной оболочки в файле `/etc/passwd` может открыть хакеру «потайную дверь», и администратор не заметит этой трансформации. Но после выполнения команды `lastlog` все неявное становится явным.

Обращайте внимание на тип подключения и адрес. Если что-то вызывает подозрение, то можно выявить атаку на этапе ее созревания.

lsof

С помощью команды `lsof` можно определить, какие файлы и какими пользователями открыты в текущий момент. Результат ее выполнения приведен в листинге 12.2.

Листинг 12.2. Результат выполнения команды `lsof`

COMMAND	PID	USER	FD	TYPE	DEVICE	SIZE	NODE	NAME
init	1	root	cwd	DIR	3,2	4096	2	/
init	1	root	rtd	DIR	3,2	4096	2	/
init	1	root	txt	REG	3,2	26920	635256	/sbin/init
init	1	root	mem	REG	3,2	89547	553856	/lib/ld-2.2.5.so
init	1	root	10u	FIFO	3,2		195499	/dev/initctl
keventd	2	root	cwd	DIR	3,2	4096	2	/
keventd	2	root	rtd	DIR	3,2	4096	2	/
kapmd	3	root	10u	FIFO	3,2		195499	/dev/initctl

Это далеко не полный результат. Даже если вы сейчас один работаете с системой, количество открытых файлов может исчисляться парой десятков, и число их заметно растёт, если в системе несколько пользователей, — ведь один файл может открываться несколько раз каждым из них. Это касается в основном системных конфигурационных файлов.

12.4.2. Системные текстовые журналы

Следующие журналы — это текстовые файлы. Их можно без проблем просматривать такими командами, как `cat`, или любыми текстовыми редакторами.

/etc/shadow, если получить права администратора. Но даже в этом случае вы почувствуете неладное, когда увидите запись о том, что в систему вошел пользователь, которого вы не создавали.

Чтобы реагировать на подозрительные записи, вы должны быть внимательны. Самые опытные и крайне опасные хакеры используют очень интересные методы. Например, в вашей системе есть пользователь robert. Хакер видит эту запись и добавляет пользователя с именем rodert. Разница всего лишь в третьей букве, и если быть невнимательным, то ее и не заметишь, что позволит взломщику безнаказанно гулять по вашему компьютеру.

Журнал почтового сервера sendmail находится в файле /var/log/maillog. В нем можно увидеть строки примерно следующего вида (приведенные здесь три строки представляют собой одну строку журнала):

```
Jan 16 13:01:01 FlenovM sendmail[1571]: j0GA11S01571: from=root,  
size=151, class=0, nrcpts=1,  
msgid=<200501161001.j0GA11S01571@flenovm.ru>, relay=root@localhost
```

Из этого файла вы можете узнать, кто, когда и кому отправлял сообщения.

Вспоминается случай, когда один администратор после того, как взломали его сервер, «вручную» просматривал все каталоги на предмет поиска чужих файлов. Не проще ли проанализировать журнал, где все записано? Если злоумышленник не подчистил журналы до выхода из системы, то можно узнать достаточно много.

На журналы надеяться можно, но все же это не очень надежно. Умные хакеры всегда заматают свои следы и уничтожают ненужные записи из журналов. Поэтому и «ручной осмотр» может пригодиться, но первым делом стоит проверить журнал.

12.4.3. Журнал FTP-сервера

Войдя в систему, взломщики нередко закачивают на сервер собственные программы для повышения своих прав или для открытия потайных дверей. Для загрузки можно использовать протокол FTP. Кто именно подключался к серверу, можно узнать из файла /var/log/secure. А вот что закачивали — подскажет /var/log/xferlog.

Журнал FTP-сервера текстовый, как и у почтового сервера, но мы его рассматриваем отдельно. Из моей практики: если вы используете сервис FTP, то именно он чаще всего приводит к проблемам. Нет, программа достаточно хороша, но злоумышленник, как правило, стремится получить доступ к учетной записи с правами на FTP, чтобы иметь возможность размещать на сервере свой «боевой софт». С помощью анализа журнала вы быстро узнаете, кто и что закачивал.

Посмотрим на строку из файла /var/log/xferlog (здесь опять длинная строка разбита на две):

```
Sun Jan 16 13:21:28 2005 1 192.168.77.10 46668 /home/flenov/sendmail.cf  
b _ o r flenov ftp 0 * c
```


12.4.4. Журнал прокси-сервера squid

Основным журналом прокси-сервера squid является `/var/log/squid/access.log`. Это текстовый файл, в котором каждая строка состоит из следующих полей:

- ☐ время начала соединения или события;
- ☐ продолжительность сессии;
- ☐ IP-адрес клиента;
- ☐ результат обработки запроса. Здесь может быть одно из следующих значений:
 - `TCP_HIT` — в кэше найдена нужная копия;
 - `TCP_NEGATIVE_HIT` — объект кэширован негативно, получена ошибка при его запросе;
 - `TCP_MISS` — объект не найден в кэше;
 - `TCP_DENIED` — отказ в обслуживании запроса;
 - `TCP_EXPIRED` — объект найден, но устарел;
 - `TCP_CLIENT_REFRESH` — запрошено принудительное обновление;
 - `TCP_REFRESH_HIT` — при попытке обновления сервер сообщил, что объект не изменился;
 - `TCP_REFRESH_MISS` — после попытки обновления сервер вернул новую версию объекта;
 - `TCP_REF_FAIL_HIT` — объект из кэша устарел, а новую версию получить не удалось;
 - `TCP_SWAPFAIL` — объект должен находиться в кэше, но он не найден;
- ☐ количество байтов, полученных клиентом;
- ☐ метод запроса — `GET`, `POST`, `HEAD` или `ICP_QUERY`;
- ☐ URL-адрес запрашиваемого объекта;
- ☐ поле `ident` (знак «минус»), если оно недоступно;
- ☐ результат запроса к другим кэшам:
 - `PARENT_HIT` — объект найден;
 - `PARENT_UDP_HIT_OBJECT` — объект найден и возвращен в UDP-запросе;
 - `DIRECT` — объект запрошен с оригинального сервера;
- ☐ тип содержимого MIME.

Когда в *главе 9* мы говорили о прокси-сервере squid, то упоминали и о других журналах: `cache.log` и `useragent.log`.

делать. Если сейчас вы еще не прочувствовали необходимость использования журналов, то после первой проблемы или взлома системы вы осознаете все их преимущества.

Программа `syslogd` сохраняет в файлах журналов всю информацию о сообщениях системы. Программа `klogd` предназначена для сохранения сообщений ядра. Настройки журналов находятся в файле `/etc/syslog.conf`. Пример содержимого файла можно увидеть в листинге 12.3.

Листинг 12.3. Файл конфигурации программы `syslogd`

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
# Выводить все сообщения ядра на экран.
# Вывод других сообщений создаст на экране беспорядок.
#kern.*                               /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
# Протоколировать в указанный файл все сообщения уровня info и выше.
# Исключения составляют письма, сообщения аутентификации и демона cron.
*.info;mail.none;authpriv.none;cron.none    /var/log/messages

# The authpriv file has restricted access.
# Файл authpriv содержит ограниченный доступ.
authpriv.*                                   /var/log/secure

# Log all the mail messages in one place.
# Сохранять все события почтовой системы в указанное место.
mail.*                                       /var/log/maillog

# Log cron stuff.
# Протоколировать сообщения cron.
cron.*                                       /var/log/cron

# Everybody gets emergency messages.
# Все получают критические сообщения.
*.emerg                                     *

# Save news errors of level crit and higher in a special file.
# Сохранять сообщения новостей уровня crit (критический) и выше
# в специальный файл.
uucp,news,crit                             /var/log/spooler

# Save boot messages also to boot.log.
# Сохранять сообщения, происходящие во время загрузки в указанный файл.
local7.*                                   /var/log/boot.log
```


просмотрит конфигурационный файл `/etc/syslog.conf` и найдет новое расположение журналов.

Однако мы можем поступить еще умнее, и для этого достаточно штатных средств ОС Linux. В моей системе сервис `cron` раз в час запускает процесс, который делает резервную копию каталога `/var`. Таким образом, если хакер подчистит журнал, я легко смогу определить его по резервной копии.

Если у вас есть возможность установить в сети еще один Linux-сервер (специально выделенный компьютер), то можно все сообщения журнала отправлять на него, что будет еще более выгодно. Чтобы хакер смог подправить журнал, ему придется взламывать и этот сервер тоже. А если он используется только для протоколирования и никаких лишних портов на нем не открыто, взлом его может оказаться затруднительным.

Для журналирования по сети в файле `/etc/services` должна иметься строка:

```
syslog 514/udp
```

Кроме того, в файле `/etc/syslog.conf` должна присутствовать директива:

```
сообщения @адрес
```

Первый параметр указывает, какие сообщения нужно отправлять на сервер. Если вы хотите пересылать всю информацию, то в качестве этого параметра указывается `*.*`, если только критические — то `*.crit`.

Второй параметр — это адрес сервера, на который будут отправляться сообщения журналов. Например, если вы хотите, чтобы все сообщения отправлялись на сервер `log.myserver.com`, то добавьте строку:

```
*.* @log.myserver.com
```

Но тут есть одна проблема — для определения IP-адреса необходим DNS. При загрузке системы сервис `syslogd` стартует раньше DNS, поэтому определение адреса окажется невозможным. Чтобы решить эту задачу, соответствие IP-адреса и имени сервера можно прописать в файле `/etc/hosts`.

И последнее. На сервере служба `syslogd` должна быть запущена с ключом `-r`, который позволяет получать сообщения по сети и сохранять их в журнале. Для этого нужно изменить сценарий запуска сервиса. Все сценарии хранятся в каталоге `/etc/rc.d/init.d/`, и для `syslogd` это файл `syslog`, основное содержимое которого можно увидеть в листинге 12.4.

Листинг 12.4. Содержимое файла `/etc/rc.d/init.d/syslog`

```
#!/bin/bash

. /etc/init.d/functions

[ -f /sbin/syslogd ] || exit 0
[ -f /sbin/klogd ] || exit 0
```


Здесь в кавычках указываются параметры. Чтобы добавить ключ `-r`, измените строку следующим образом:

```
SYSLOGD_OPTIONS="-m 0 -r"
```

Если файл `/etc/sysconfig/syslog` существует, то в нем будет такая же строка. В этом случае вам достаточно внести изменения в этот файл, и не надо будет трогать сценарий запуска `/etc/rc.d/init.d/syslog`.

Использование выделенного сервера для сохранения сообщений из журналов позволяет сохранить записи, но при этом делает их доступными для всеобщего просмотра. Дело в том, что сообщения передаются по сети в незашифрованном виде. Это не проблема, если вы отделены от Интернета сетевым экраном, и злоумышленник не смог проникнуть внутрь сети. Но если хакеру удалось взломать хотя бы один компьютер в защищенной сети, то он может установить программу прослушивания и увидеть все сообщения в журналах.

Вопрос решается достаточно просто — зашифровать трафик, направив его через туннель SSL. Самый простой вариант — выполнить следующие действия:

1. В конфигурационном файле сервера, отправляющего сообщения, прописываем пересылку сообщений на локальный компьютер:

```
*.* @localhost
```

Все сообщения будут передаваться на порт 514 UDP локального компьютера. Чтобы все получилось, сервер не должен работать в режиме приема сообщений, т. е. запущен без ключа `-r`. Иначе порт 514 будет занят, а нам это не нужно.

2. На порту 514 локального компьютера запускаем `stunnel`-клиент:

```
stunnel -c -d 127.0.0.1: 514 -r logserver:1050
```

Все сообщения, полученные на этот порт, будут шифроваться и передаваться на порт 1050 компьютера `logserver`. В вашем случае вместо имени `logserver` нужно указать адрес вашего сервера.

3. На компьютере `logserver` создаем `stunnel`-сервер:

```
stunnel -d 1050 -r 127.0.0.1:514
```

После этих действий на локальном компьютере клиент `stunnel` будет получать незашифрованные сообщения на порту 514, шифровать их и отправлять на порт 1050 компьютера `logserver`. А на компьютере `logserver` сервер `stunnel` будет получать зашифрованные сообщения и в расшифрованном виде направлять их на порт 514. На сервере `logserver` сервис `syslogd` должен быть запущен с ключом `-r` для приема сообщений на порту 514.

Теперь все сообщения будут передаваться по сети в зашифрованном виде.

12.4.7. Утилита `logrotate`

Чтобы файлы журналов слишком сильно не раздувались, в Linux включена утилита `logrotate`. Рассмотрим принцип ее работы на примере журнала `/var/log/messages`:


```
rotate 1
}

# system-specific logs may be also be configured here.
# Специфичные системные журналы могут быть сконфигурированы здесь.
```

Посмотрим на параметры, которые нам доступны:

- ❑ `weekly` — указывает на то, что файлы журналов должны меняться еженедельно. Если сервер используется редко, можно изменить это значение на `monthly`, чтобы обновление происходило ежемесячно;
- ❑ `rotate` — задает количество файлов, которые будут использоваться для хранения истории. В нашем случае стоит число 4, т. е. имена журналов будут иметь вид: `/etc/log/имя.X`, где `X` изменяется от 1 до 4;
- ❑ `create` — требует создания нового пустого документа после смены файла журнала;
- ❑ `compress` — позволяет сжимать файлы журналов. На серверах, обрабатывающих запросы огромного количества пользователей, журналы могут занимать очень много места, и чтобы сэкономить дисковое пространство, их можно сжимать. Так как журналы содержат текст, сжатая версия может иметь объем на 70 и более процентов меньше.

В начале файла идут описания значений по умолчанию. Затем можно указать специфичные значения для определенных журналов. В нашем конфигурационном файле выделен журнал `/var/log/wtmp`:

```
/var/log/wtmp {
    monthly
    create 0664 root utmp
    rotate 1
}
```

В файле нет ограничения на размер журнала, но его можно задать с помощью параметра `size`. Например, в следующем примере задается максимальный размер журнала в 100 Кбайт:

```
/var/log/wtmp {
    monthly
    size = 100k
    create 0664 root utmp
    rotate 1
}
```

Теперь файл журнала будет меняться в двух случаях (по событию, которое наступит раньше):

- ❑ ежемесячно, т. к. указан параметр `monthly`;
- ❑ когда файл достигнет размера 100 Кбайт.

Конечно же, профессиональные взломщики, которые зарабатывают деньги, проникая в чужие системы, знают об этом, поэтому делают все возможное, чтобы скрыть свои действия, и регулярно чистят этот журнал. Посторонние изменения файла `.bash_history` могут указать на конкретную учетную запись, через которую произошел взлом.

Пользовательские журналы нужно регулярно проверять и чистить. Некоторые пользователи (да и вы сами) могут ошибиться при написании какой-либо команды и указать свой пароль. Взломщик, проанализировав файл `.bash_history`, увидит пароли и сможет уничтожить ваш сервер.

Если вы в командной строке набирали директиву и указывали пароль администратора `root`, то не поленитесь удалить соответствующую запись из пользовательского журнала. Такая ошибка может вам обеспечить бессонную ночь и не исключено, что не одну.

Пароли администраторов могут указываться в командной строке и при использовании программы `mysql`. Если вы, например, выполнили команду:

```
/usr/bin/mysql -uroot -ppassword
```

то она полностью сохранится в журнале. Получив доступ к вашему журналу команд `bash`, злоумышленник приобретает возможность использовать базу данных MySQL с правами `root`. В лучшем случае это будет только база данных, а в худшем (если пароль `root` на систему и на MySQL совпадают), весь сервер будет под контролем взломщика.

ВНИМАНИЕ!

Никогда не выполняйте в командной строке директивы, требующие указания пароля, а если сделали это, то удалите соответствующую запись из журнала `bash`. В случае с MySQL нужно было задать команду `/usr/bin/mysql -uroot`. В ответ на это сервер запросит пароль, который не сохранится в журнале, а запишется только введенная команда.

Если вы работаете с сервером баз данных MySQL, то в вашем домашнем каталоге помимо файла `.bash_history` будет еще и файл `.mysql_history`. В этом файле хранятся все команды, которые выполнялись в программе конфигурирования `mysql`. Его также надо очищать, если при выполнении команды был указан какой-либо пароль. База данных — это еще не вся ОС, но и она может послужить отправной точкой для дальнейшего взлома, к тому же сами базы данных могут содержать секретную информацию — например, пароли доступа к закрытым частям веб-сайта.

12.4.9. Обратите внимание!

Давайте посмотрим, на чем нужно сосредоточить внимание при анализе журналов безопасности. Это не только записи о неправомерном доступе к системе. Наблюдая за журналами, необходимо выявлять атаки еще на начальном этапе и не допускать взлома. Когда вы увидели, что пользователь получил доступ к закрытой информации, то момент взлома вы уже пропустили.

Теперь представим, что хакер накидал строк, в которых поле `logname` всегда разное. Выделить из них реальные практически невозможно.

Хорошо, если при выполнении программы `logger` хакер не будет использовать ключ `-t`, а укажет команду следующим образом:

```
logger -p kern.alert "authentication failure ; logname=robert uid=0 \
euid=0 tty=tty1 ruser= rhost= user=root "
```

В этом случае в журнале появится строка:

```
Feb 12 17:31:37 FlenovM logger: authentication failure; logname=robert
uid=0 euid=0 tty=tty1 ruser= rhost= user=root
```

Как видите, перед сообщением об ошибке стоит ключевое слово `logger`, которое как раз и выдает подделку.

Даже если программа `logger` не будет доступна хакеру, он может создать записи своей программой.

12.5. Работа с журналами

Мы рассмотрели различные журналы, которые доступны в системе, взглянули на их содержимое и узнали, что и где хранится. Знать все это просто необходимо, но анализировать текстовый файл размером в несколько мегабайт очень сложно и неудобно.

В действующей системе, обрабатывающей множество пользовательских запросов, журналы растут очень быстро. Например, на моем веб-сервере ежедневный журнал часто превышает 4 Мбайт, а ведь мой сайт — далеко не самый посещаемый. Это очень много текстовой информации, в которой быстро найти нужную строку практически невозможно.

Для упрощения анализа программисты и администраторы написали и продолжают разрабатывать программы-анализаторы файлов журналов. Просматривать журналы необходимо каждый день, а лучше даже каждый час. Для обеспечения безопасности нельзя прозевать важные сообщения, иначе проигрыш будет обеспечен. Но как при ежечасном контроле файла отделить записи, которые уже проверялись? Программа должна уметь запоминать время последней ревизии и при следующем запуске отбрасывать ранее просмотренные записи.

Наиболее эффективными программами анализа журналов являются сервисы, которые проверяют записи параллельно с попаданием их в `log`-файлы. Это достаточно просто реализовать, особенно на удаленном компьютере, которому по сети посылается информация от работающего сервера. По мере получения записей они проверяются и помещаются в файлы для дальнейшего хранения и более тщательного анализа. По одной записи очень сложно выявить атаку, и иногда необходимо видеть динамику событий. Например, одна неправильная попытка авторизации еще ни о чем не говорит, а вот 10 и более — это уже должно вызывать подозрение.

У меня несколько сайтов, и HTTP-трафик всех записывается в один и тот же файл журнала `http80`. Чтобы отсеять только нужный мне домен, я использую `grep`. Теперь нужно подождать несколько секунд и посмотреть, идет ли какой-нибудь трафик через кэш-сервер (появляются ли в журнале новые записи).

12.5.2. Программа `swatch`

Программа `swatch` — очень мощная утилита, написанная на простом и знакомом многим администраторам языке Perl. Это позволяет легко изменять программу и добавлять к ней возможности самостоятельно. Программа позволяет анализировать записи по расписанию (если занести выполнение программы в `cron`) или сразу после их попадания в журнал. Скачать программу можно по адресу: sourceforge.net/projects/swatch.

Так как это Perl-программа, то процесс ее установки отличается от многих других программ. В нашем случае выполните следующие действия:

```
tar xzvf swatch-3.1.tgz
cd swatch-3.1
perl Makefile.PL
make test
make install
make realclean
```

Как всегда, у этой медали есть обратная сторона. То, что программа написана на Perl, является и недостатком, поскольку требует наличия интерпретатора. Я уже говорил, что нельзя устанавливать на сервер то, что может открыть дверь злоумышленнику и при этом не является действительно необходимым. Без нужды я рекомендую не подключать интерпретаторы типа Perl, потому что хакеры очень часто используют их для написания собственных rootkit-программ.

12.5.3. Программа `Logsurfer`

Одна из немногих программ, которая может просматривать журнал в динамике, — `Logsurfer` (sourceforge.net/projects/logsurfer). Как я уже говорил, большинство средств просматривает журнал построчно, что не очень эффективно, потому что в результате мы получаем слишком много мусора.

Из-за мощных возможностей этой программы ее сложнее настраивать. Это тоже недостаток, потому что из-за ошибок в ее конфигурации можно не уловить очень важные события.

12.5.4. Программа `Logcheck/LogSentry`

Это самая простая в использовании программа. В состав `LogSentry` уже входят различные шаблоны, с помощью которых можно выделять потенциально опасные записи.

в вашей системе в эту группу входит только администратор, то можно оставить все как есть. Но если в нее входит несколько пользователей, что я абсолютно не приветствую, то необходимо сформировать специальную группу с минимальными правами и назначить ее для всех журналов.

Следующие команды создают новую группу `loggroup` и устанавливают ее для всех `log`-файлов:

```
groupadd logsgroup  
cd /var/log  
chgrp -R logsgroup .
```

В каталоге `/var/log` правом чтения и записи в журналы должен обладать исключительно администратор. Пользователям группы следует разрешить только чтение, а остальным — запретить абсолютно все. Для того чтобы всем файлам установить эти права, выполните следующие команды:

```
cd /var/log  
find . -type f | xargs chmod 640
```

Вторая строка состоит из двух директив. Команда `find . -type f` ищет в текущем каталоге все объекты, у которых тип равен `f`, т. е. все файлы. Вторая (`xargs chmod 640`) — изменяет у всех найденных объектов права доступа на 640. Можно даже понизить это значение до 600, чтобы и читать, и писать мог только администратор.

Помимо этого, для каталога `/var/log` у пользователей не должно быть прав на запись, потому что это позволит злоумышленнику удалять файлы. Если хакер не сможет изменить журнал, то попытается его уничтожить. Да, вы поймете, что в системе кто-то был, но не определите, как произошло вторжение, и не сможете найти взломщика.

Помните, что, прочитав журнал, хакер может получить шанс повысить свои права, если там случайно окажется конфиденциальная информация. Но если журналы станут доступными на запись, то взломщик сможет замести следы, удалив все записи относительно своей активности.

Но и этого недостаточно для обеспечения максимальной защиты. Если посмотреть на суть журналов, то станет очевидным, что в них ОС только добавляет новые записи. Соответственно, можно поставить дополнительную защиту от удаления и изменения записей на основе специальных ключей. В файловых системах Ext2, Ext3 и Ext4 есть команда `chattr`, с помощью которой можно устанавливать на файлы дополнительные атрибуты. Один из них (ключ `+a`) позволяет задать условие, при котором файл можно только пополнять. Например, следующая команда устанавливает для файла `/var/log/boot.log` такой атрибут:

```
chattr +a /var/log/boot.log
```

Попробуйте теперь изменить или удалить файл. У вас ничего не выйдет. Единственный недостаток этого ключа — у вас тоже не будет возможности чистить файл. А ведь журнал постоянно растет, и нет смысла хранить записи о событиях, которые



ГЛАВА 13

Резервное копирование и восстановление

Те из вас, кто долго работает в сфере информационных технологий, уже не раз сталкивались с проблемой потери данных. Но мы продолжаем уделять этому вопросу слишком поверхностное внимание.

Многие считают, что аппаратура сейчас надежна, и из-за своей лени никогда не делают резервных копий. Техника хороша, но на моих глазах «умерло» уже несколько винчестеров, украдено из офисов пять компьютеров, полностью сгорел вместе с кабинетом один сервер. А кто из работавших в великих башнях города Нью-Йорка думал, что к ним прилетят самолеты? Кто-то скажет, что такие слова безжалостны, ведь произошла трагедия. Но мы тоже не застрахованы от жестоких действий террористов, и Россия уже сталкивалась с этим. И как бы ни было больно, закрывать на это глаза нельзя. Необходимо делать все, чтобы данные были сохранены в любой ситуации.

Резервное копирование — сохранение всех важных файлов в другом каталоге или на отдельном носителе. Если регулярно осуществлять такую операцию, то в случае непредвиденной ситуации есть возможность восстановить файлы из резервной копии и продолжить работу с незначительными потерями.

На macOS и Windows я делаю резервные копии довольно редко, потому что там все пользовательские данные можно хранить в «облаке», что я и делаю. «Облака» OneDrive и iCloud дают высокую гарантию, что файлы никуда не денутся. В Linux, наверно, самым лучшим вариантом может быть Dropbox. Но это защита пользовательских данных, а как насчет серверных?

13.1. Основы резервного копирования

Чтобы понять, что резервное копирование необходимо, давайте рассмотрим основные ситуации, когда оно помогает:

- *случайное изменение или удаление файлов* — когда к серверу подключается новый пользователь, не имеющий достаточного опыта работы с компьютерами,

очень часто его нелепые действия приводят к уничтожению данных. При правильной политике безопасности могут быть разрушены только его собственные файлы, но и они могут иметь ценность для организации;

- *нарушение работы устройств* — когда я только начинал знакомиться с компьютерами, то в обиходе были дискеты 5,25 дюйма и жесткие диски максимальным размером в 20 Мбайт. Если винчестеры были относительно надежны, то информация на дискетах постоянно пропадала из-за порчи поверхности носителя. С переходом на дискеты 3,5 дюйма ситуация изменилась не сильно, а вот надежность жестких дисков со временем повышалась еще больше. Но когда мы начали оперировать гигабайтами, то в определенный момент я реально столкнулся с проблемой испорченных блоков (Bad Blocks). Примерно за полгода мне пришлось сменить три жестких диска разных производителей размером от 10 до 20 Гбайт. Это было как набег саранчи, уничтожающей посевы. В настоящее время надежность дисков снова достигла высокого уровня, но ее нельзя назвать идеальной. Всегда есть вероятность, что диск выйдет из строя;
- *стихийные бедствия и потеря техники* — не будем больше говорить о терроризме, потому что есть и другие причины утраты техники. Если оглянуться на последние годы, то замечаешь, что наша планета начинает преподносить страшные сюрпризы. Я имею в виду участвовавшие наводнения, смерчи, землетрясения и пожары, которые могут уничтожать дома, офисы и целые города. Если есть резервная копия и она хранится в другом городе (в этом поможет Интернет) или в негорящем сейфе, то восстановить данные не составит труда;
- *хакеры и эпидемии вирусов* — как же без них... Это порождение информационной жизни, без которого уже никуда не деться, и приходится выстраивать всевозможные оборонительные рубежи. Какое средство чаще всего используют для защиты от вредоносного кода? Конечно же, антивирусы. Они запрещают выполнение любого известного злого кода. Но хакеры придумывают новые программы и способы обхода антивирусов. И именно новые вирусы наносят максимальный ущерб, потому что для них нет еще эффективного метода лечения. Потери от эпидемий с каждым годом увеличиваются, но их можно сократить с помощью резервного копирования и восстановления.

Этот список можно продолжать еще долго, но, надеюсь, я смог вас убедить в необходимости иметь резервную копию всей значимой информации. Благодаря использованию «облачных» дисков я стал делать резервные копии намного реже, но все же раз в несколько месяцев я банально создаю ZIP-архив всей своей рабочей папки и копирую ее на внешний диск. Это что касается личных данных.

Посмотрим теперь, что именно нужно копировать. К важным данным можно отнести:

- *системные конфигурационные файлы* — на первый взгляд это не так важно, потому что в таких файлах нет секретной и важной для организации информации. Но конфигурирование ОС и всех программ с чистого листа потребует гораздо больше времени, чем восстановление с использованием резервных файлов. А долгое время восстановления влечет за собой потери из-за простоя. На Linux я

Помимо повышения надежности работы, кластер может увеличивать и производительность, если все серверы функционируют параллельно и нагрузка распределена равномерно.

Еще более дешевым вариантом является использование резервных дисков. Допустим, у вас есть один сервер, который должен быть всегда доступен пользователям. В нем устанавливается дисковый массив RAID (Redundant Array of Independent Disks, избыточный массив независимых дисковых накопителей) с поддержкой зеркалирования (Mirroring) — т. е. RAID 1 или RAID 1+0. В этом случае RAID заботится о сохранности данных, т. к. их запись производится на два диска одновременно. Если один из них сломается, то полная копия сохранится на другом.

А что, если выйдет из строя материнская плата или процессор? Их замена потребует времени, а мы договорились, что это недопустимо. Чтобы сократить простой в случае такой нештатной ситуации, подготавливаем заранее сервер с идентичной конфигурацией оборудования. При нарушении работы «железа» достаточно перенести RAID-массив на резервный сервер, переключить сетевой кабель, и можно продолжать трудиться. Поскольку оборудование на обоих компьютерах одинаковое, переустановка системы не потребуется, и RAID будет работать на другом сервере без необходимости изменения конфигурационных файлов.

Если в вашей сети несколько серверов с одинаковой конфигурацией, то один резервный компьютер может служить заменой любого из них. Обеспечение сохранности данных таким способом — намного дешевле построения кластера.

В одном офисе я видел очень интересное решение. На всех клиентских компьютерах были установлены маленькие жесткие диски, на которых работала только ОС и необходимые ей файлы и программы. Помимо этого, у всех через Mobile Rack (устройство, позволяющее сделать винчестер съемным) были подключены большие диски. Администратор каждый вечер снимал эти диски и делал резервные копии на своем компьютере.

Такой подход позволяет при возникновении проблем с железом или ОС снять жесткий диск и перенести его на другой компьютер. Для этого у грамотного администратора всегда есть подготовленный системный блок, которым можно заменить испорченное оборудование. Но с таким же успехом кто-то может взять жесткий диск и унести его домой, а на работе оставить другой диск, который будет выглядеть так же. Не думаю, что администратор будет проверять серийный номер каждого из дисков.

13.3. Хранение резервных копий

Несмотря на использование RAID и кластера, резервное копирование никто не отменял, и его делать необходимо. Но куда резервировать данные? Однажды меня вызвали восстановить данные после выхода из строя жесткого диска. Воскресить информацию, конечно же, не удалось, потому что винчестер вышел из строя окончательно и бесповоротно, поэтому я задал вполне логичный вопрос: «А где резервная копия?» Ответ был прост (впрочем, как и владельцы компьютера) — запасная

- максимальное время простоя (недоступности) сервера;
- категория наиболее часто меняющихся данных.

Этот список можно продолжить, но мы пока остановимся и обсудим его. Нужно четко разобраться, какие данные в системе изменяются. После этого следует разделить их на три группы в зависимости от периодичности модификации: часто, редко и с определенным интервалом.

Вот основные каталоги, которые должны резервироваться:

- `/etc` — содержит конфигурационные файлы;
- `/home` — пользовательские файлы;
- каталог, содержащий веб-файлы.

В остальных каталогах редко хранятся документы или необходимые для сохранения файлы. Программы из каталогов `/bin` или `/usr` нет смысла дублировать, потому что их легко переустановить, особенно если сохранены настройки.

13.4.1. Редко, но метко...

К нечасто изменяемым файлам можно сразу отнести конфигурационные файлы (каталог `/etc`). В этом каталоге массовые корректировки происходят на этапе установки сервера. Затем компьютер может работать годами, и изменения производятся в случае обновления программ или внесения каких-то поправок.

Для хранения конфигурации хватит даже самого небольшого носителя с невысокой скоростью. Единственное требование к нему — должна иметься возможность перезаписи. Я для этих целей сейчас использую диски, подключаемые по USB.

Поскольку конфигурация изменяется редко, то можно делать копии сразу после внесения правок. Для этого достаточно записать на резервный диск измененный файл и не копировать все остальные конфигурационные файлы.

При восстановлении данных необходимо всегда начинать с конфигурации — в первую очередь с файлов `/etc/passwd` и `/etc/shadow`. Если этого не сделать, то ни одна программа не сможет установить правильные права доступа.

13.4.2. Зачастили...

Часто изменяемыми могут быть базы данных и основные файлы и документы пользователей (каталог `/home`), которые корректируются каждый день. Их резервные копии можно и нужно создавать ежедневно. Если процесс копирования отнимает слишком много времени, то следует это делать по окончании рабочего дня или в обеденный перерыв, когда нагрузка на сервер ниже. Чтобы не сидеть над компьютером в такие моменты, можно создать сценарии, которые будут выполняться по расписанию. Если производить резервирование два раза в день (в обеденный перерыв и в конце рабочего дня), то в случае аварии вы рискуете потерять изменения только за полдня (с момента резервирования до сбоя системы).

регулярно их резервную копию не имеет смысла. Намного эффективнее делать это в конце отчетного периода, а потом весь месяц не тратить ресурсы на лишние операции с неизменяемыми данными.

13.4.5. Полная копия...

Наиболее надежным способом является создание полной копии всего жесткого диска. В этом случае информация может сохраняться независимо от файловой системы, потому что программа копирует весь диск (один к одному), используя прямой доступ к дорожкам. Восстановление полной копии гарантирует, что все права настроены четко, и программы сразу же готовы к использованию.

Но этот способ имеет достаточно много недостатков:

- ☐ требует много времени;
- ☐ вызывает слишком большую нагрузку;
- ☐ может включать в резервную копию все файлы, даже те, которые не являются необходимыми, например каталог `/tmp`.

Резервирование полной копией очень удобно использовать для переноса данных на другой сервер или тиражирования конфигурации. Например, вам необходимо настроить несколько однотипных клиентских компьютеров. Сконфигурируйте один, сделайте его полную копию и восстановите на остальных машинах. Такой метод надежнее, чем простой перенос файлов с одного компьютера на другой.

13.5. Резервирование в Linux

Мы рассмотрим только те возможности, которые уже реализованы в Linux. Я не буду рекламировать сторонние разработки. Я вообще не люблю кого-то выделять, потому что могу недооценить другие продукты, с которыми я не работал или не знаком в достаточной степени.

А что встроено в Linux? Это простые команды копирования и архивирования, которые можно автоматизировать, прописав в планировщике задач. Если резервирование требует выполнения нескольких директив, то из них можно создать файл сценария, который будет выполнять все необходимое.

13.5.1. Копирование

Самый простой вариант создания резервной копии — использование команды `cp` (копирование файлов). Только при копировании необходимо обязательно сохранять права доступа к файлу. Вот как может выглядеть команда, сохраняющая каталог `/home` на примонтированный к системе диск `/mnt/resdisk`:

```
cp -a /home /mnt/resdisk
```

Здесь я использовал ключ `-a`, который равносителен указанию сразу трех ключей (`-dpr`):


```
cd /
tar xf /home/backup.tar
```

Здесь мы из корневого каталога разархивируем резервную копию, которая находится в файле /home/backup.tar.

Помимо этого, для архивных операций могут пригодиться следующие ключи утилиты tar:

- ☐ **v** — вывести на экран информацию об архивируемом (или распаковываемом) в данный момент файле;
- ☐ **z** — найти и обработать при распаковке GZIP-архивы;
- ☐ **p** — разархивировать всю информацию о безопасности;
- ☐ **d** — найти различия между архивом и файлами системы;
- ☐ **t** — просмотреть содержимое архива;
- ☐ **u** — обновить файлы в архиве, которые были изменены;
- ☐ **n date** — добавить в архив только те файлы, которые изменены позже указанной даты. Параметр date должен быть заменен необходимой датой;
- ☐ **P** — не удалять первый символ /. В этом случае, в каком каталоге вы ни запустите разархивирование, файлы попадут на свое родное место.

С помощью утилиты tar можно архивировать сразу несколько каталогов. Следующая команда помещает в архив каталоги /home и /etc:

```
tar cf backup.tar /home /etc
```

Чтобы просмотреть содержимое архива, можно выполнить команду:

```
tar tvf backup.tar
```

В ответ на это на экране будут показаны все файлы и каталоги архива, их права доступа и владельцы. Результат можно увидеть в листинге 13.1.

Листинг 13.1. Результат просмотра содержимого архива

```
drwx----- 504/504          0 2004-11-27 20:24:05 home/adr/
drwxr-xr-x 504/504          0 2004-11-27 20:24:05 home/adr/.kde/
drwxr-xr-x 504/504          0 2004-11-27 20:24:05 home/adr/.kde/share/
-rw-r--r-- 504/504        118 2004-11-27 20:24:05 home/adr/.gtkrc
-rw-r--r-- 504/504         24 2004-11-27 20:24:05 home/adr/.bash_logout
-rw-r--r-- 504/504        191 2004-11-27 20:24:05 home/adr/.bash_profile
-rw-r--r-- 504/504        124 2004-11-27 20:24:05 home/adr/.bashrc
-rw-r--r-- 504/504          5 2004-11-27 20:24:05 home/adr/text
-rw-r--r-- 504/504       2247 2004-11-27 20:24:05 home/adr/.emacs
```

Посмотрите на последнюю колонку, где показано расположение файла. Обратите внимание, что путь не начинается с символа /, указывающего на корень диска. Поэтому такой архив нужно распаковывать в корне, иначе он будет создаваться в текущем каталоге.

времени на архивирование окажутся идентичны затратам на копирование без архивирования или даже меньше.

Прежде чем сжимать какой-либо файл, рекомендуется подготовить TAR-архив. Потом достаточно выполнить команду упаковки:

```
gzip -уровень файл.tar
```

В качестве ключа `-уровень` нужно указать степень компрессии. Максимальный уровень равен 9. После этого указывается имя TAR-архива. Давайте сожмем архивный файл, который мы создали из каталога `/home`, применяя наибольшую компрессию. Выполните следующую команду:

```
gzip -9 backup.tar
```

Теперь просмотрите содержимое каталога (команда `ls`). Обратите внимание, что файла `backup.tar` больше нет. Вместо него появился файл `backup.tar.gz`, размер которого значительно уменьшился.

Чтобы разархивировать такой файл, можно пользоваться все той же командой `tar`, только необходимо указать ключи `xfz`:

```
cd /  
tar xzf /home/backup.tar.gz
```

Эта команда сначала разархивирует GZ-файл и тут же распакует TAR-архив.

Если необходимо из GZ-файла снова получить TAR-архив (без его распаковки), то можно выполнить команду:

```
gzip -d /home/backup.tar.gz
```

После этого вы снова увидите файл `backup.tar`, а файл `backup.tar.gz` исчезнет.

Теперь вы готовы написать свой сценарий, который будет собирать каталоги для архивирования в TAR-файл, а затем сжимать его, чтобы уменьшить размер. Но можно не использовать две команды, а обойтись одной. Вот как это делается:

```
tar cvf - /home | gzip -9c > backup.tar.gz
```

Здесь мы собираем в TAR-архив каталог `/home` и тут же сжимаем его утилитой `gzip`.

Помимо `gzip` для архивирования иногда используется утилита `compress`, но ее возможности по сжатию ниже, и к тому же вокруг нее возникали скандалы и разбирательства по поводу лицензии. Большинство администраторов уже перешли на использование `gzip`, и я вам рекомендую с самого начала привыкать к этой программе.

13.5.4. Утилита `dump`

Все предыдущие команды, которые мы рассматривали в этой главе, не являются специализированными командами резервирования. Это просто команды копирования и архивирования файлов. А вот утилита `dump` предназначена именно для создания резервной копии файловой системы Ext (поддерживаются версии, начиная со второй).

Однажды я видел, как секретные данные с хорошо защищенного сервера каждый час копировались на простой компьютер пользователя, на котором все настройки были установлены по умолчанию. Такую систему хакер взломает за пять минут.

К защите резервных копий нужно подходить со всей ответственностью. Самый простой вариант — поместить их в сейф. Но лучше будет зашифровать файл перед записью резервных копий на носитель. Напоминаю, что сделать это (для примера с файлом `backup.tar.gz`) можно с помощью пакета `OpenSSH`, используя следующую команду:

```
/usr/bin/openssl des -in /home/backup.tar.gz -out /home/backup.sec
```

В ответ на это будет создан файл `backup.sec`. Именно его и надо записывать на носитель для долгосрочного хранения. Только не забудьте удалить потом с диска файлы `backup.tar.gz` и `backup.sec`.

При восстановлении файл сначала необходимо расшифровать:

```
/usr/bin/openssl des -d -in /home/backup.sec -out /home/backup.tar.gz
```

После этого можно разархивировать все файлы на свои места.

13.7. Облака

Все больше популярности набирают «облачные» технологии. Самыми известными провайдерами сейчас являются `Amazon` и `Microsoft` — обе компании предоставляют хорошие и удобные средства для построения кластеров и обеспечивают великолепную отказоустойчивость.

Я пользуюсь `Amazon`, и мне нравятся виртуальные машины `Linux`, которые доступны администраторам для использования в «облаке». «Облачные» технологии хоть и выходят за рамки книги, но мне хотелось упомянуть, что они существуют и в «облаках» тоже можно создавать `Linux`-серверы и работать с ними, как с любой другой машиной на этой ОС.



ГЛАВА 14

Советы на прощанье

Здесь собраны некоторые советы на различные темы. Создавать главу ради каждого из этих советов просто не имеет смысла, поэтому я собрал их все под одной шапкой. Надеюсь, что вам поможет эта информация.

В первом издании книги в этой главе был описан взлом, теперь же я перенес ту информацию на свой сайт www.flenov.info в раздел **Статьи**.

14.1. Пароли

Не буду повторять, что пароли должны быть сложными, а обращаю ваше внимание на другое. Все пароли необходимо менять через определенные периоды времени. Если честно, то я ненавижу этот процесс, но у нас на работе политикой безопасности предусмотрено менять пароли каждые три месяца. И несмотря на то, что меня этот процесс бесит, я понимаю, что он необходим.

Многие хакеры, получая доступ к системе, некоторое время не проявляют никакой активности. Они осматриваются, знакомятся с принципами работы системы и определяют, как сделать, чтобы их не вычислили. Быстрых действий можно ожидать только от того, кто проникает в систему ради уничтожения всех данных и кому нет смысла замечать следы своего пребывания. Слава богу, таких взломов не так уж и много.

Итак, проникнув в систему, хакер будет незаметно сидеть в ней, и вы можете ничего не заподозрить. Но если каждый месяц меняется пароль, то после очередной смены злоумышленник теряет свои права, и ему требуется повторный взлом для выявления нового пароля. Если уязвимость, через которую он узнал пароль какого-то пользователя, уже залатана, то второй раз получить доступ ему будет сложнее.

Регулярное обновление паролей усложняет также и их подбор. Как это происходит? Многие автоматизированные системы выявления атак могут без проблем определить, когда на отдельную учетную запись авторизуются несколько раз подряд. Чтобы обойти такие системы, хакеры проверяют пароли с определенной задержкой. Это делает взлом дольше, но в конце концов может дать результат, если пароль

несложный и постоянный. Если пароль изменяется, то вероятность успеть его подобрать до смены становится очень низкой.

Чтобы увидеть это на примере, представим, что пароль может содержать только числа. Допустим, что на первоначальном этапе он был равен 7 000 000. Хакер τυ-пым перебором прошел от 0 до 6 000 000, и в этот момент пароль меняется на 5 000 000. Дальнейшее сканирование хоть до миллиарда не даст результата, потому что диапазон, в котором находится новый пароль, уже пропущен. Хуже будет, если новый пароль будет как раз одним из следующих в переборе. Тут уж не угадаешь.

Второе преимущество от регулярной смены пароля заключается в том, что, пока хакер будет подбирать действительно сложный пароль, он уже устареет и воспользоваться им не удастся.

Как заставить пользователя менять пароли через определенные промежутки времени? Нет, не нужно ходить и нудить ему над ухом — есть способ проще и намного эффективнее. В Linux имеется утилита `chage`, которая запускается следующим образом:

`chage` параметры пользователь

В качестве параметра можно указывать следующие ключи:

- ❑ `-m N` — минимальное число дней (N) до смены пароля. Указав это значение чуть меньше, чем максимальный период (см. следующий параметр), вы защитите систему от нежелательной смены паролей. Это значит, что если хакер захватит учетную запись, он не сможет изменить пароль. Конечно же, злоумышленник тоже может выполнить команду `chage`, но только если у него есть права администратора. Три-четыре дня разницы между минимальным и максимальным значением необходимы для того, чтобы пользователь смог сменить пароль, пока он не устарел. Устанавливать разницу меньше трех дней нежелательно, потому что существуют выходные, и если срок действия попадет на воскресенье, пользователь не успеет сменить пароль. По умолчанию используется значение `-1`, что соответствует отсутствию проверки;
- ❑ `-M N` — максимальный диапазон в днях (N), в течение которого действует пароль. После этого пароль считается недействительным, и пользователь не сможет войти в систему. По умолчанию установлено `99999`, что соответствует бесконечности, а значит, пароль никогда не устареет;
- ❑ `-d N` — дата последнего изменения пароля. Параметр N указывает количество дней начиная с 1 января 1970 года. Если установить `1000`, то получится 27 сентября 1972 года. Чтобы не высчитывать дату в днях, можно указать ее явным образом в формате ГГГГ-ММ-ДД;
- ❑ `-E дата` — дата окончания действия пароля;
- ❑ `-I N` — период в днях, после которого неиспользуемая учетная запись блокируется. Рекомендую указать не менее 3 и не более 4 дней, чтобы приостановить действие записи на время отпуска или болезни работника;

Чтобы включить модуль `pam_cracklib.so`, необходимо добавить в файл `/etc/pam.d/passwd` следующую строку:

```
password required pam_cracklib.so retry=5 minlength=8
```

Этой командой мы заставляем систему использовать библиотеку `pam_cracklib.so`. Параметр `retry` задает число попыток для ввода нового пароля, а они понадобятся, если пользователь попытается задать слишком простую комбинацию. Параметр `minlength` задает минимальную длину пароля.

Все вроде бы прекрасно, но у такой политики есть и недостатки. Как показывает моя практика, большинство пользователей выбирают один пароль и в конце добавляют число 1. При следующей смене пароля добавляется число 2. И так далее. Поскольку операционные системы не хранят реальных паролей, а только хеши, то они не могут видеть, что новый пароль изменился всего на одну цифру, и что-либо предпринять. Так что в реальности даже требование сменить пароли не помогает. Да что там скрывать, я сам так поступаю регулярно.

14.2. rootkit: «набор администратора»

Проникнув в систему, хакер стремится укрепиться в ней и получить максимальные права. Пусть он уже может выполнять на сервере команды от имени простого пользователя. Этого ему будет мало, поэтому следующая цель — получение прав `root` со всеми вытекающими отсюда последствиями.

Для решения этой задачи взломщик должен получить возможность закачивать файлы и установить в системе одну из специализированных программ, повышающих права до администратора, — такие программы называются `rootkit` («набор администратора»). После этого взломщик выполняет команды следующим образом:

1. От имени простого пользователя, правами которого обладает хакер, директивы посылаются программе `rootkit`.
2. Программа `rootkit` выполняет полученные команды от имени администратора.

А как же `rootkit` получает возможность выполнять переданные ей команды с правами `root`? Тут может использоваться уязвимость в ОС или помогает злополучный SGID-бит. Если он установлен, то программа будет выполняться в системе с правами администратора.

Но, к счастью, не все так просто. Для `rootkit` нужно еще установить SGID-бит и в качестве владельца установить пользователя `root`.

Тут есть два пути:

- если есть возможность выполнять команды `chown` и `chmod`, то хакер сможет без проблем реализовать все необходимые действия;
- можно подменить программу на ту, которая уже имеет установленный SUID-или SGID-бит.

Вы должны быть внимательны, когда проверяете SGID-программы. Хакеры знают, что администраторы стараются свести количество таких программ к минимуму,

- ❑ `-sS` — TCP SYN-сканирование. Если вы работаете с правами `root`, то по умолчанию установлен этот тип, как более быстрый и к тому же неопределяемый некоторыми программами антисканирования;
- ❑ `-sF` — TCP FIN-сканирование. В соответствии с RFC 793, если на порт направить пакет с установленным флагом `FIN` (используются для завершения соединения) и этот порт окажется закрытым, то сервер должен ответить пакетом, имеющим тип `RST`. ОС Linux действует по стандарту, и поэтому можно легко просканировать порты с помощью этого метода. Если пакет `RST` не получен, то порт открыт. А вот работа Windows далека от стандарта, и здесь результат непредсказуем;
- ❑ `-sX` — TCP Xmas-сканирование. Метод похож на предыдущий, только помимо этого устанавливаются флаги `URG` и `PUSH`, указывающие на срочность данных;
- ❑ `-sN` — TCP NULL-сканирование. На сервер направляются пустые пакеты, на которые он должен сообщить об ошибке;
- ❑ `-I` — Ident-сканирование;
- ❑ `-sU` — UDP-сканирование.

Смысл сканирования в том, чтобы получить от сервера хоть какой-нибудь ответ. В зависимости от метода сканирования по положительному или отрицательному ответу определяется, закрыт порт или открыт.

Более быстрый способ получить открытые порты — это команды `lsof` (с параметром `-i`) или `netstat`, но их выполнение должно происходить локально, непосредственно с компьютера. Вторая директива будет эффективной только в том случае, если хакер в текущий момент подключен к системе.

Помимо `rootkit`, вы должны проверить систему на наличие посторонних загружаемых модулей ядра. Для этого очень хорошо подходит утилита `chkproc` (входит в состав пакета `chkrootkit`). Но и это еще не все, пакет `chkrootkit` включает в себя еще и утилиту `ifpromisk`, которая позволяет найти программу прослушивания трафика.

И напоследок нужно проверить список работающих процессов с помощью команды:

```
ps -aux
```

чтобы найти незнакомые процессы. При просмотре будьте внимательны. Вспомните пример с программой `login`, когда первая буква `l` заменялась цифрой `1`. Быстрым взглядом скользнув по процессу `login`, можно ничего не заметить.

Определив наличие файлов `rootkit`, вы должны остановить их работу и удалить из системы. Самый простой вариант, если программа хакера не модифицировала никаких системных файлов. Если же это произошло, то нужно переустановить все программы, которые изменил злоумышленник. Легче всего это сделать в дистрибутивах на основе Red Hat, где поддерживается работа с RPM-пакетами. Тогда достаточно выполнить команду:

```
rpm -U -force пакет.rpm
```

Здесь мы запрашиваем восстановление пакета `пакет.rpm`.

подключения, — это использовать команду `netstat`. Но поскольку эта программа входит в состав Linux, то ее исходные коды также могут быть изменены. А вот от сканера портов не скроешься; правда, для его работы необходимо больше времени.

Но и от сканера портов `backdoor` может скрыться — точнее, он может вовсе не открывать портов. Лучший способ спрятать `backdoor` от сетевых анализаторов — использовать при программировании Raw Sockets (сырые сокеты), как это делают sniffеры. На сервере программа `backdoor` прослушивает весь трафик, и если видит пакеты, помеченные специальным образом, то выполняет инструкции, описанные в этих пакетах. Хакеру только остается направлять широковещательные или просто безымянные пакеты, имеющие определенный идентификатор, чтобы сервер выполнял необходимые инструкции.

Утилита `netstat` и сканеры портов не могут определить sniffеры, поэтому они тут бессильны. Однако для прослушивания трафика сетевая карта должна работать в специализированном режиме, который легко определяется, если просмотреть состояние сетевого интерфейса командой `ifconfig`.

14.4. Небезопасный NFS

Технология NFS (Network File System, сетевая файловая система) была разработана компанией Sun Microsystems в 1989 году. Идея была великолепной. Любой пользователь может монтировать каталоги сервера к своей файловой системе и использовать их, как будто они находятся на компьютере клиента. Это очень удобно в сетях. Пользовательские каталоги могут находиться на сервере и подключаться к клиенту по мере надобности. Таким образом, все файлы будут храниться централизованно, а использоваться, как будто они находятся локально.

Но, как я уже говорил, удобство и безопасность — несовместимые вещи, а NFS слишком удобна.

В состав NFS входит утилита `showmount`, которая может отобразить, какие каталоги и какими пользователями подключены. Для администратора это неоценимая информация.

Выполните команду:

```
showmount -a localhost
```

Вы увидите информацию о NFS на своем сервере в таком формате:

```
All mount points on localhost:
robert:/home/robert
econom:/home/jhon
buh:/home/andrey
robert:/usr/local/etc
econom:/usr/games
```

Результат разделен на две колонки символами двоеточия. В первой находится имя компьютера, подключившего удаленный раздел, а во второй — путь на сервере к подключенному ресурсу.

конкретным пользователям, которые действительно монтируют свои файловые системы удаленно. Например:

```
/home/Robert          rw
/home/FlenovM          rw
/home/Andrey           rw
```

Большинство специалистов по безопасности сходятся во мнении, что NFS не стоит использовать вообще. Если вы решили применить ее только для того, чтобы программы были установлены централизованно, то следует победить свою лень и заняться их установкой на каждый компьютер в отдельности.

Если вам необходимо сделать документы общедоступными, чтобы пользователи могли работать совместно с одним каталогом, то можно рассмотреть вариант использования Samba (см. главу 6). Этот сервис менее болтлив и может решить ваши потребности в разделении каталогов сервера.

14.5. Определение взлома

Для эффективной защиты сервера очень важно вовремя определить, что сервер был взломан. Чем раньше вы узнаете о проникновении в систему хакера, тем скорее сможете отреагировать и предотвратить печальные последствия. Помните, взломы бывают, ломали даже именитые компании, потому что людям свойственно ошибаться. Тут важно правильно реагировать на инцидент.

14.5.1. Осведомлен — значит защищен

Ранее я очень часто использовал чрезвычайно эффективный, но сложный в реализации метод — информирование при запуске потенциально опасных программ. Сложность заключается в том, что надо уметь программировать под Linux хотя бы на каком-нибудь языке программирования. Лучше, если это будет C, но можно и Perl. В крайнем случае подойдет умение писать сценарии (командные файлы).

Итак, в чем заключается мой метод? Войдя в систему, хакер всегда оглядывается и старается найти способ укрепиться в системе, чтобы оставаться долгое время незаметным для администратора. Для этого взломщик чаще всего выполняет команды `who`, `su`, `cat` и др. Ваша задача — установить на них ловушки. Например, можно изменить код программы `su` так, чтобы сразу после ее выполнения администратору направлялось письмо.

Получение сообщения о том, что была выполнена опасная команда, если она запускалась не администратором, — хороший повод проверить систему на наличие в ней постороннего.

Если вы не умеете программировать, можно обойтись и средствами самой ОС. Допустим, что вы хотите получать сообщения каждый раз, когда выполняется команда `who`. Взломщик часто выполняет такую директиву, когда входит в систему, чтобы

Таким образом можно подменить все опасные программы, которые должны быть недоступны простым пользователям.

Хакеры чаще всего не проверяют утилиты, которые запускают, а ведь угрозу можно увидеть, если выполнить команду:

```
cat /usr/bin/who
```

Вот тут и проявляется недостаток использования сценариев — их можно просмотреть. А программы, написанные на языке С и откомпилированные в исполняемый файл, при просмотре показывают абсолютно ничего не говорящий мусор.

14.5.2. Ловля на живца

Администраторы очень любят оставлять приманки для хакеров, потому что они позволяют идентифицировать злоумышленника на начальном этапе. Эта технология даже получила название honeypot («горшок меда»). Как это работает? В сети устанавливается один или несколько компьютеров, в которых изначально заложены ошибки конфигурирования или легкие для подбора пароли. Основная задача этого комплекса — отслеживание обращений извне и регистрация любых взломов.

На рис. 14.1 приведена схема классической honeypot-сети. От Интернета ее отделяет сетевой экран, за которым находятся публичные ресурсы и подставные серверы/компьютеры (публичная сеть). Далее идет второй сетевой экран, который защищает и скрывает приватную сеть.

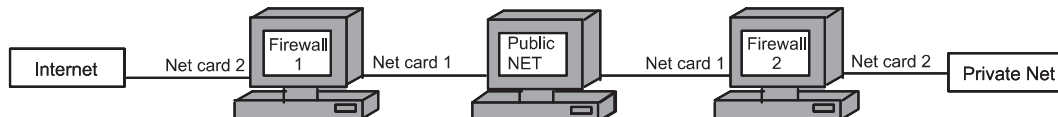


Рис. 14.1. Построение honeypot-сети

Как только хакер «попадает в капкан», администраторы начинают его идентификацию и поиск. Пока злоумышленник пытается проникнуть далее, в хорошо защищенную сеть, специалисты по безопасности уже успевают прийти к нему домой и физически остановить нездоровое любопытство.

Чтобы ваш «капкан» не ловил всех подряд и не давал ложных срабатываний, его защита должна быть достаточной, чтобы сервер нельзя было сломать программами, автоматизирующими поиск уязвимостей. Иначе количество ежедневно пойманных в ловушку хакеров будет исчисляться сотнями, а то и более, ведь популярные ресурсы сканируют часто.

Если честно, то я уже давно не пользуюсь ловушками, но решил все же о них рассказать. Мое отношение к ним изменилось, и больше я не считаю их эффективным и заслуживающим внимания средством. Но, возможно, вы найдете в них что-то полезное для себя, поэтому раздел о них остался в этом издании.

Я раньше настраивал свои honeypot-серверы на максимальную безопасность, просто сетевой экран, который их защищает, пропускал практически любой трафик. Это позволяло решить сразу несколько потенциальных проблем:

Каждый лишний модуль — это очередной удар по производительности и безопасности. Поэтому их нужно отключить. Закончив с этим, переходим к более тонким настройкам.

14.6.1. Параметры ядра

Для начала откроем конфигурационный файл `/etc/sysctl.conf`. В нем находятся параметры ядра. Пример файла можно увидеть в листинге 14.1.

Листинг 14.1. Конфигурационный файл `/etc/sysctl.conf`

```
# Kernel sysctl configuration file for Red Hat Linux
# Конфигурационный файл ядра для Red Hat Linux

# For binary values, 0 is disabled, 1 is enabled.
# See sysctl(8) for more details.
# Для бинарных значений 0 — это отключен, а 1 — включен.
# Смотрите man sysctl для получения дополнительной информации

# Controls IP packet forwarding
# Контролирует переадресацию IP-пакетов
net.ipv4.ip_forward = 0

# Controls source route verification
# Контроль проверки маршрутизации от источника
net.ipv4.conf.default.rp_filter = 1

kernel.sysrq = 1

kernel.core_uses_pid = 1

#net.ipv4.tcp_ecn = 0

kernel.grsecurity.fifo_restrictions = 1
kernel.grsecurity.linking_restrictions = 1

# audit some operations
# Аудит некоторых операций
kernel.grsecurity.audit_mount = 1
kernel.grsecurity.signal_logging = 1
#kernel.grsecurity.suid_logging = 1
kernel.grsecurity.timechange_logging = 1
kernel.grsecurity.forkfail_logging = 1
kernel.grsecurity.coredump = 1

# lock all security options
# Блокировка всех опций безопасности
#kernel.grsecurity.grsec_lock = 1
```


- ❑ `net.ipv4.conf.*.secure_redirects` — позволить принимать от шлюза по умолчанию сообщения о перенаправлении маршрутизатора. Параметр может быть включен, только если в вашей сети действительно более одного маршрутизатора, иначе лучше его запретить;
- ❑ `net.ipv4.conf.*.send_redirects` — разрешить компьютеру, если он является маршрутизатором, отправлять сообщения о перенаправлении маршрутизатора. Если в сети несколько маршрутизаторов, то параметр можно включить, чтобы распределить нагрузку между ними и не пытаться пропускать весь трафик через основной шлюз;
- ❑ `net.ipv4.conf.*.accept_source_route` — позволить принимать пакеты с маршрутизацией от источника. В *разд. 14.7.2* мы еще обсудим этот вопрос, а сейчас необходимо знать, что такие пакеты могут стать причиной обхода вашего сетевого экрана. Запретите этот параметр;
- ❑ `net.ip_always_defrag` — дефрагментировать все приходящие пакеты. Так уж повелось, что сетевой экран проверяет только первый пакет, а все остальные считает разрешенными. Хакер может обойти сетевой экран, прибегая к разбивке посылки на части (см. *разд. 14.7.1*). Если установить этот параметр, то все входящие пакеты будут дефрагментированы и обход сетевого экрана этим методом станет невозможным;
- ❑ `net.ipv4.ipfrag_low_thresh` — определяет минимальный объем памяти, который выделяет ОС для сборки фрагментированных пакетов. Чем больше это значение, тем меньше будет манипуляций по выделению памяти. По умолчанию используется число 196 608. Слишком большое значение отнимет лишнюю память и может привести к эффекту, при котором для обработки данных не хватит ресурсов сервера. Я бы оставил значение по умолчанию;
- ❑ `net.ipv4.ipfrag_high_thresh` — определяет максимальное количество памяти (по умолчанию 262 144), выделяемое для сборки фрагментированных пакетов. Если значение превышено, то ОС начинает отбрасывать пакеты. Хакер может попытаться закидать сервер большим количеством мусорных сообщений, но сервер больше не будет реагировать на фрагментацию;
- ❑ `net.ipv4.ipfrag_time` — определяет время хранения фрагментированных пакетов в кэше. По умолчанию используется значение 30 секунд. Это очень много, за это время хакер сможет забросать весь кэш. В случае атаки на систему следует понизить это значение до 20, а то и до 10 секунд;
- ❑ `net.ipv4.tcp_syncookies` — продолжая тему защиты от DoS, я рекомендую включить этот параметр, чтобы защититься от атаки SYN Flood, при которой на сервер направляется большое количество пакетов с запросом на соединение. Хакер устанавливает в пакетах ложный обратный адрес, и сервер ожидает соединения от несуществующих или ничего не подозревающих компьютеров. Таким образом легко превышает максимально допустимое количество подключений.

Параметров ядра очень много, и рассматривать все мы не будем. В принципе всю необходимую информацию можно узнать из документации.

Как видите, мы просто привели все ключи и указали диск `/dev/hda`. Обратите внимание, что при определении устройства не стоит никаких цифр, которые указывали бы на раздел, т. к. доступ можно изменить только жесткому диску в целом.

После изменения параметров их необходимо сохранить с помощью команды:

```
hdparm -k1 /dev/hda
```

После этого снова выполните команду тестирования скорости работы диска:

```
hdparm -t /dev/hda
```

Есть еще один параметр, который влияет на производительность, — режим доступа. В настоящее время поддерживаются три режима АТА: 33/66/100. Сверьтесь с документацией на жесткий диск, чтобы узнать, что он поддерживает. Для смены режима служит ключ `-X`:

□ `-X34` — АТА33;

□ `-X68` — АТА66;

□ `-X69` — АТА100.

Для установки АТА66 выполните команду:

```
hdparm -X68/dev/hda
```

Странно, но установленные вами параметры не сохраняются после перезагрузки системы, поэтому желательно прописать эти команды в файл `/etc/rc.d/rc.local`. Для этого в самый конец файла добавляем три строки:

```
hdparm -m128d1c3/dev/hda
```

```
hdparm -X68/dev/hda
```

```
hdparm -k1 /dev/hda
```

14.6.3. Автомонтирование

Если вы начали знакомство с компьютером под ОС Windows, то вам покажется дикостью процесс ручного монтирования файловых систем и особенно компакт-дисков. Действительно, для сервера это еще приемлемо, потому что там диски используются редко, а вот на рабочей станции внешние носители применяются регулярно. Мне иногда приходится вставлять по 20 разных дисков в день, и каждый раз монтировать и демонтировать их очень неудобно.

Так как ОС Linux все больше поворачивается в сторону домашних пользователей — в последних дистрибутивах производителями по умолчанию включена возможность автоматического монтирования. Для этого используется служба `autofs`. Убедитесь, что она запускается у вас, и можно приступать к настройке.

Основной конфигурационный файл сервиса `autofs` — файл `/etc/auto.master`. Просмотрите его содержимое в листинге 14.2.


```
#floppy          -fstype=ext2          :/dev/fd0
#e2floppy        -fstype=ext2          :/dev/fd0
#jaz             -fstype=ext2          :/dev/sdc1
#removable       -fstype=ext2          :/dev/hdd
```

Теперь рассмотрим содержимое файла `/etc/auto.misc`. Здесь только одна строка без комментария, которая описывает команды подключения компакт-диска:

```
cd              -fstype=iso9660,ro,nosuid,nodev      :/dev/cdrom
```

Первый параметр определяет каталог внутри `/misc`, куда будет монтировано устройство. Второй атрибут — параметры файловой системы и ключи, которые будут применяться для подключения. В случае с CD-ROM используется файловая система ISO9660 и опции, разрешающие только чтение и запрещающие запуск SUID- и SGID-программ. Последний параметр определяет устройство, которое должно монтироваться.

Как видите, все очень даже просто. Если попытаться обратиться к каталогу `/misc/cd` и в приводе CD-ROM в этот момент будет находиться диск, то он окажется автоматически смонтирован. Правда, для этого нужно использовать команды Linux (например, выполнить команду `ls /misc/cd`), а не другие программы. Если же попытаться просмотреть каталог `/misc/cd` с помощью Midnight Commander, то диск смонтирован не будет.

14.7. Короткие советы

Мы проанализировали достаточно много аспектов создания безопасной системы, но есть некоторые общие рекомендации, подытоживающие рассмотренный в этой книге материал. Поэтому напоследок я собрал короткие советы, которые пригодятся вам при построении безопасного сервера или сети.

14.7.1. Дефрагментация пакетов

С помощью фрагментированных пакетов хакеры производят очень много атак на серверы. В Linux можно сделать так, чтобы ОС объединяла приходящие пакеты. Если у вас монолитное ядро (без поддержки модулей), то необходимо прописать 1 в файл `/proc/sys/net/ipv4/ip_always_defrag`. Это легко сделать с помощью команды:

```
echo 1 > /proc/sys/net/ipv4/ip_always_defrag
```

В последних ядрах, которые используют RPM-модули, необходимо подгрузить модуль `ip_conntrack`:

```
modprobe ip_conntrack
```

14.7.2. Маршрутизация от источника

Мы уже говорили о том, как пакеты проходят по сети. Напомню, что внутри сети пакеты передаются по MAC-адресу, а для обмена между сетями необходимо мар-

ных случаях. Но необходимо сначала убедиться, что вы работаете именно со второй или более старой версией и что шифрование используется.

14.7.4. Полный путь

Когда вы запускаете какие-либо команды или программы, то необходимо указывать полный путь к ним. Большинство пользователей и администраторов просто указывают имя запускаемого объекта, что может стать причиной взлома. Да что там говорить, я сам грешу вводом коротких команд.

Рассмотрим пример того, как злоумышленник может использовать короткие имена в своих целях на примере команды `ls`:

1. Хакер создает в каком-либо каталоге (например, в общедоступном `/tmp`) файл с таким же именем, как и у атакуемой программы.
2. В этот файл хакер записывает сценарий, который выполняет необходимые ему действия.
3. В переменную окружения `PATH` добавляется путь к этому каталогу.

Например, в файл может быть записан следующий код:

```
#!/bin/sh
# Изменяем права доступа к файлам /etc/passwd и /etc/shadow
chmod 777 /etc/passwd > /dev/null
chmod 777 /etc/shadow > /dev/null
# Выполняем программу ls
exec /bin/ls "$@"
```

В этом примере выполняются всего три команды. Первые две изменяют права доступа к файлам `/etc/passwd` и `/etc/shadow` так, чтобы любой пользователь смог их прочитать. При этом все сообщения, которые могут возникнуть во время выполнения команд, направляются на нулевое устройство `/dev/null`, чтобы они не отображались на экране. После этого выполняется системная команда `ls` из каталога `bin`.

Теперь устанавливаем программе права, которые позволят выполнять ее любому пользователю:

```
chmod 777 /tmp/ls
```

Ложный файл готов. Далее необходимо сделать так, чтобы он выполнялся вместо системной команды `ls`. Для этого достаточно добавить каталог `/tmp` в самое начало системной переменной окружения `PATH`. Если теперь кто-то запустит команду `ls` без указания полного пути, то выполнится наш сценарий, который попытается изменить права доступа на файлы паролей. Если у пользователя, выполнившего команду, хватит полномочий для изменения прав, то можно считать, что система взломана.

Следите за содержимым системной переменной окружения `PATH`, чтобы ее никто не изменил.

Заключение

Какой бы надежной ни была программа или ОС и какой бы надежной ни были конфигурация и окружающие защитные комплексы, в системе все равно остается одно очень слабое звено — человек. Вы можете ставить какие угодно сложные пароли, но если кто-то из администраторов захочет продать информацию, то он это сделает. Если администратор несерьезно относится к безопасности, то его пароли могут подобрать или украсть с помощью вредоносных программ.

Безопасность — это невероятно сложный и комплексный подход. Кто-то пытается решить его введением бюрократии и различных правил работы с ИТ, вытаскивает CD-приводы из компьютеров и опечатывает USB-разъемы. Отделы безопасности начинают разрабатывать политики безопасности, которые ограничивают свободу сотрудников. Очень часто такое поведение вызывает раздражение со стороны опытных пользователей, и особенно программистов. По своему опыту знаю, как они не любят, когда им что-то запрещают подобными способами.

Чтобы уменьшить вероятность возникновения конфликтных ситуаций и недопонимания, желательно защищать только то, что действительно требует защиты. Невозможно защитить абсолютно все.

Человек может быть самым слабым звеном в системе, а может оказаться и самым сильным. А для этого, на мой взгляд, просто нужно хорошо изучить матчасть и руководящие материалы по используемой ОС.

ПРИЛОЖЕНИЕ 1

Команды протокола FTP

Когда вы подключаетесь к FTP-серверу с помощью клиента, работающего из командной строки (например, `sftp`, упомянутого в *разд. 5.3.7*), то для работы с сервером вам помогут следующие команды:

- ☐ `cd путь` — изменить текущий каталог на указанный. Чтобы подняться на один уровень выше, можно выполнить команду `cd ..`, а чтобы перейти в папку ниже текущего уровня — команду `cd каталог`;
- ☐ `bye` — разорвать соединение;
- ☐ `exit` — выйти из системы;
- ☐ `chmod права имя файла` — изменить права доступа к файлу. Например, чтобы установить права доступа `770` на файл `passwd` в текущем каталоге, нужно выполнить команду `chmod 770 passwd`;
- ☐ `get -R удаленный_файл локальный_файл` — скачать файл. Ключ `-R` является необязательным, он позволяет сохранить права доступа на файл в локальной системе, сделав их такими же, как и на сервере. Эта опция не работает, если файл передается между разными системами, потому что в Windows совершенно другой метод хранения прав доступа. Параметр `локальный_файл` указывает на полный путь к файлу, где должен быть сохранен скачанный с сервера файл;
- ☐ `put -R локальный_файл удаленный_файл` — закачать локальный файл на сервер. Выполнение команды схоже с `get`, только в данном случае локальный файл закачивается на сервер;
- ☐ `help` — отобразить список команд, которые разрешено выполнять;
- ☐ `pwd` — отобразить текущий каталог;
- ☐ `rm файл` — удалить файл;
- ☐ `rmdir каталог` — удалить каталог;
- ☐ `mkdir имя` — создать каталог с указанным именем.

Вы должны учитывать, что разные FTP-серверы и FTP-клиенты могут обрабатывать команды по-разному.

ПРИЛОЖЕНИЕ 2

Полезные программы

Эти программы могут помочь вам в борьбе с хакерами. Сперва укажем снифферы, т. е. утилиты для прослушивания трафика:

□ **dsniff (monkey.org/~dugsong/dsniff)** — пакет программ для прослушивания трафика, который состоит из следующих утилит:

- **dsniff** — служит для перехвата паролей. Она прослушивает трафик в ожидании пакетов авторизации, и если такой пакет найден, то заветный пароль выводится на экран. Поддерживается поиск пакетов авторизации всех основных протоколов, таких как TELNET, FTP, POP и т. д.;
- **arp spoof** — позволяет отправлять ARP-ответы, с помощью которых можно обмануть компьютеры жертвы, сказав, что определенный IP принадлежит вам;
- **dnsspoof** — позволяет отправлять поддельные DNS-ответы. Если жертва запрашивает IP-адрес сервера, вы можете подделать ответ DNS-сервера, чтобы вместо нужного ему сервера клиент подключился к вашему компьютеру и вы смогли перехватить на себя трафик;
- **filesnaf** — прослушивает трафик в ожидании передачи файлов по NFS;
- **mailsnaf** — прослушивает сеть в ожидании E-mail-сообщений по протоколам POP и SMTP;
- **msgsnaf** — отслеживает сообщения интернет-пейджеров и чатов, таких как ICQ и IRC;
- **masof** — позволяет наводнить сеть пакетами со сгенерированными MAC-адресами. Если коммутатор перестает справляться с нагрузкой по определению пути, то он начинает работать как простой концентратор, и вы сможете прослушивать трафик всех компьютеров сети;
- **tcpkill** — может завершить чужое соединение, отправив поддельный пакет с установленным флагом RST;
- **webspy** — позволяет прослушивать соединения с веб-серверами и сохраняет список сайтов, которые посещал определенный пользователь;

- webmint — эмулирует веб-сервер и становится посредником. Технология атаки этим методом рассмотрена в *разд. 7.9*;

❑ ettercap (ettercap.sourceforge.net) — на мой взгляд, это самая удобная программа для прослушивания трафика. Основное назначение программы — поиск паролей в пакетах всех популярных протоколов. Администраторы оценят возможность программы определять наличие в сети других программ подслушивания трафика.

Теперь напомним программы, которые помогают анализировать конфигурацию сервера и определять попытки взлома:

- ❑ Isat (usat.sourceforge.net) — программа проверки конфигурации системы (см. *разд. 12.3*). Анализирует конфигурацию сервера, отображая потенциальные ошибки, и в некоторых случаях может дать рекомендации по устранению недочетов;
- ❑ bastille (bastille-linux.sourceforge.net) — система выявления потенциальных ошибок в конфигурации сервера. Программа может автоматически исправлять ошибки и недочеты в конфигурации;
- ❑ Klaxon (www.eng.auburn.edu/users/doug/second.html) — программа для определения атак на вашу систему;
- ❑ PortSentry (sourceforge.net/projects/sentrytools) — утилита для слежения за портами и отслеживания попыток их сканирования. Позволяет автоматически сконфигурировать сетевой экран для запрета соединения с компьютером, с которого происходило сканирование;
- ❑ Swatch (sourceforge.net/projects/swatch) — удобная программа анализа журналов по расписанию (см. *разд. 12.5.2*);
- ❑ Logsurfer (sourceforge.net/projects/logsurfer) — одна из немногих программ, позволяющих анализировать журнал безопасности в динамике (см. *разд. 12.5.3*).

Напоследок упомянем еще парочку полезных программ:

- ❑ John the Ripper (www.openwall.com/john) — самая знаменитая программа подбора паролей;
- ❑ Nmap (nmap.org) — сканер портов, который обладает большим количеством возможностей.

ПРИЛОЖЕНИЕ 3

Интернет-ресурсы

Я приведу лишь несколько сайтов, которые кажутся мне наиболее информативными. Вы, вероятно, расширите этот список сайтами, интересными и полезными именно для вас.

- ❑ **www.redhat.com** — сайт компании Red Hat, где можно скачать последние версии программ, ядра, патчи и прочитать о перспективах развития этой ОС.
- ❑ **www.kernel.org** — сайт, посвященный ядрам ОС Linux. Здесь же можно скачать последнюю версию ядра.
- ❑ **www.securityfocus.com** — сайт, посвященный безопасности, содержит множество описаний различных уязвимостей и методов исправления ошибок.
- ❑ **www.cert.org** — еще один сайт, посвященный информационной безопасности.
- ❑ **www.securitylab.ru** — русскоязычный сайт по информационной безопасности, где можно прочитать об уязвимостях на русском языке.
- ❑ **www.xakep.ru** — сайт российского журнала «Хакер».
- ❑ **www.insecure.org** — множество полезной информации по безопасности, статьи и программы.
- ❑ **www.suse.com** — сайт SUSE, одного из самых простых и удобных дистрибутивов Linux.
- ❑ **www.ubuntu.com** — сайт самого популярного дистрибутива, согласно статистике посещения, которую я просматривал.
- ❑ **www.debian.org** — официальный сайт дистрибутива Debian.
- ❑ **www.slackware.com** — сайт дистрибутива Slackware.

ПРИЛОЖЕНИЕ 4

Работа в командной строке

Небольшой секрет — когда вы набираете команды, то можно экономить время с помощью клавиши <Tab>. Нужно всего лишь начать набирать команду и нажать <Tab>. Например, если вы находитесь в корне, то там только один каталог на букву h: home. Наберите букву h и нажмите клавишу <Tab> — оболочка сама допишет полное имя каталога. То же самое касается и файлов.

Если вам нужно повторить выполненную ранее команду, то можете использовать клавишу <↑> или <↓>. После нажатия на клавишу <↑> в командной строке появится последняя введенная команда. Нажмите еще раз, и вы увидите предпоследнюю. Нажмите <↓>, и вы опять вернетесь к последней команде. Таким образом можно перемещаться по истории введенных команд. Заметьте, что не обязательно выполнять команду именно так, как она была введена ранее, можно ее подредактировать и лишь затем нажать клавишу <Enter>.

Псевдонимы

С помощью команды `alias` можно создавать псевдонимы для команд. Если вы часто выполняете какое-то действие, для чего нужно подать длинную команду с кучей параметров, то удобно создать для нее псевдоним. Например, для просмотра текущего каталога используется команда `ls`, но она показывает короткое содержимое, без прав доступа, а часто бывает необходимо увидеть именно их. Каждый раз указывать параметр `-l`, набирая `ls`, нудно, поэтому можно создать псевдоним `ll`:

```
alias ll="ls -l"
```

Теперь выполнение команды `ll` будет идентично выполнению `ls -l`. Но не торопитесь делать именно этот псевдоним в своей системе. Вполне возможно, что он уже у вас есть; по крайней мере, у меня в дистрибутиве такой псевдоним создан по умолчанию.

Перенаправление

Теперь поговорим о том, как можно направлять выходные данные команды не на экран, а, допустим, в файл. Для этого служит символ `>`. Например, выполните команду:

```
ls > outfile.txt
```

В результате выполнения этой команды содержимое каталога не будет выведено на экран, а будет сохранено в файл.

Если отобразить содержимое файла, подав команду

```
cat outfile.txt
```

вы увидите то, что было бы показано на экране после выполнения команды `ls` без перенаправления. Впрочем, не совсем так: в файле `outfile.txt` после такой команды окажется еще и его собственное имя, т. е. строка `outfile.txt`. Если же этого файла до выполнения команды с перенаправлением не было, то и на экране бы мы его не увидели.

Надо иметь в виду, что перенаправление перезаписывает файл без подтверждения, если он уже существовал. Чтобы не переписывать файл, а дописывать в его конец, надо использовать два знака `>>`:

```
ls >> outfile.txt
```

Мы можем не только выводить результат в файл, но и получать параметры из файла. Например, можно выполнить команду:

```
cat < infile.txt
```

Команде `cat` в качестве параметров будет передано содержимое файла `infile.txt`.

С помощью символа вертикальной черты `|` можно изменять стандартный поток ввода/вывода. Допустим, вы хотите узнать, когда в систему последний раз входил пользователь по имени `fenov`. Для этого можно выполнить команду `lastlog`. Но если в системе зарегистрировано 1000 пользователей, то найти нужного будет проблематично. Он может находиться в любом месте, ведь результат не отсортирован. Представьте себе, если бы телефонная книга содержала всех абонентов в неотсортированном виде! Это была бы катастрофа.

Как отсортировать вывод команды? Да очень просто, нужно направить ее вывод — в нашем случае список пользователей — команде `sort`, которая сортирует входящие данные и выводит их в отсортированном виде. А это можно сделать с помощью символа вертикальной черты:

```
lastlog | sort
```

Запуск в фоне

Если команда работает очень долго, то вы можете запустить ее на выполнение в фоновом режиме. Для этого служит символ амперсанда (`&`), который нужно поставить в конце команды. Например, у меня на работе есть OLAP-сервер (это программа для получения отчетности), который написан на Java и при запуске захва-

Предметный указатель

A

ACL 284
ADSL 189
Apache 227
◊ модули 230

B

Bad Blocks 364
bash 49
BIOS 84
BugTraq 329

C

cgi-bin 249

D

DES 189
DHCP-сервер 41
DMA 393
DoS 36
DSA 194

E

Ext2 33

F

Firewall 145
fsck 33
FTP 403

G

gcc 40
GnuPG 260
GRUB 84

H

HDD 393
Honeypot 388
HTTPS 242, 253
HTTP-сервер 227
Hub 181

I

ICMP 165
ICP 276
ICP-запрос 276
IP-адрес 176

K

Kerberos 197

L

Loki 154

M

MAC-адрес 105, 182
Mobile Rack 366
MySQL 52

Ж

- Жесткая ссылка 79
- Жесткий диск 393
- Журнал 337
- ◇ /var/log/httpd/access.log 345
- ◇ /var/log/maillog 342
- ◇ /var/log/messages 341
- ◇ /var/log/secure 341
- ◇ /var/log/squid/access.log 344
- ◇ /var/log/xferlog 342
- ◇ /var/run/utmp 337
- ◇ безопасность 359
- ◇ запись сообщений вручную 353

З

- Забытый пароль 88
- Загрузка 82
- Загрузчик 84
- Запрет пакетов 157
- Запуск программ в определенное время 98, 99

И

- Идентификатор процесса 92
- Иерархия DNS 318
- История команд 409

К

- Кластер 365
- Ключ открытый 194
- Ключи шифрования 194
- Команда
- ◇ alias 409
- ◇ at 98
- ◇ atq 99
- ◇ batch 99
- ◇ bg 92
- ◇ cat 65
- ◇ cd 66
- ◇ chage 378
- ◇ chattr 136
- ◇ chgrp 117
- ◇ chmod 115
- ◇ chown 117
- ◇ chroot 139
- ◇ cp 66, 370
- ◇ df 70
- ◇ fg 92
- ◇ find 67
- ◇ grep 69
- ◇ groupadd 122
- ◇ groupdel 124
- ◇ groupmod 124
- ◇ hdparm 393
- ◇ hostname 107
- ◇ hoststat 260
- ◇ htpasswd 241
- ◇ ifconfig 104, 105
- ◇ jobs 92
- ◇ kill 93
- ◇ last 338
- ◇ lastlog 339
- ◇ ln 79
- ◇ logger 353
- ◇ logrotate 350
- ◇ ls 64
- ◇ lsattr 136
- ◇ lsof 340
- ◇ mailq 260
- ◇ mailstats 261
- ◇ man 50
- ◇ mc 62
- ◇ mkdir 69
- ◇ modinfo 109
- ◇ modprobe 110
- ◇ mount 70
- ◇ netstat 177
- ◇ passwd 88, 125
- ◇ ping 176
- ◇ ps 93
- ◇ pwd 64
- ◇ rm 70
- ◇ rmdir 110
- ◇ shutdown 49
- ◇ su 46
- ◇ sudo 169, 299
- ◇ sysctl 391
- ◇ tac 66
- ◇ tail 357
- ◇ tar 335, 371
- ◇ top 95
- ◇ touch 75
- ◇ traceroute 164
- ◇ umask 118
- ◇ umount 73

Программа (*npod.*)

- ◇ Logsurfer 358
- ◇ md5sum 76
- ◇ Midnight Commander 62
- ◇ NetSonar 331
- ◇ nmap 381
- ◇ SATAN 331
- ◇ sendmail 261
- ◇ sftp 201
- ◇ sftp-server 201
- ◇ squid 274
- ◇ squidGuard 298
- ◇ stunnel 186
- ◇ WinSCP 202

Прокси-сервер 273

- ◇ анонимный 275
- ◇ кэш 274, 282
- ◇ прозрачный 275

Прослушивание трафика 252

Протоколы и номера портов 188

Процесс 91

Псевдонимы 409

Р

Разметка диска 28

С

Сервер: фактор конфигурации 331

Сертификат авторизации 187

Сетевой экран 145, 332

Символьная ссылка 80

Сканирование

- ◇ автоматизированное 333
- ◇ локальное 333
- ◇ методы 332
 - зондирование 332
 - имитация 332
 - сканирование 332

◇ сервера 331

◇ удаленное 332

Сниффер 405

Ссылка

- ◇ жесткая 79
- ◇ символьная 80

Т

Текстовый режим 61

Терминальный доступ 178

Тест FTP 306

Туннелирование 182, 189

Тюнинг 389

У

Удаленное управление сервером 193

Уникальный идентификатор

◇ группы 87

◇ пользователя 87

Ф

Файл

- ◇ атрибуты 135
- ◇ владелец 65, 117
- ◇ дата изменения 65, 75
- ◇ имя 78
- ◇ копирование 63, 66
- ◇ перемещение 64
- ◇ права доступа 113, 115
- ◇ ссылка 79
- ◇ удаление 137

Файлы: создание из консоли 387

Фильтрация сайтов 295

Фоновый процесс 91

Фрагментация пакетов 163

Ш

Шифрование

- ◇ трафика 186
- ◇ файлов 188, 376
- ◇ FTP 308

Э

Электронная почта 255

Я

Ядро 22

◇ параметры 390