Михаил Фленов

Linux глазами хакера

Санкт-Петербург
«БХВ-Петербург»
2005

УДК 681.3.06
ББК 32.973.26-018.2
Ф69

Фленов М. Е.

Ф69 Linux глазами хакера. — СПб.: БХВ-Петербург, 2005. — 544 с.: ил.

ISBN 5-94157-635-8

Рассмотрены вопросы настройки ОС Linux на максимальную производительность и безопасность. Описаны потенциальные уязвимости и рекомендации по предотвращению возможных атак. Дается подробное описание настройки прав доступа и конфигурирования сетевого экрана. Показано, как действовать при атаке или взломе системы, чтобы максимально быстро восстановить ее работоспособность и предотвратить потерю данных.

Для пользователей, администраторов
и специалистов по безопасности

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Игорь Шишигин*
Зав. редакцией	*Григорий Добин*
Редактор	*Алла Воробейчик*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Игоря Цырульникова*
Зав. производством	*Николай Тверских*

ISBN 5-94157-635-8

Оглавление

Предисловие

Данная книга посвящена рассмотрению одной из самых популярных операционных систем (ОС), устанавливаемых на серверы — ОС Linux. Для домашнего применения эта система пока еще не пользуется такой популярностью, как среди профессиональных администраторов, но в последнее время наметились предпосылки для захвата и этого рынка.

Установка ОС становится все проще, а графический интерфейс и удобство работы в некоторых случаях не уступает самой прославленной в среде малого бизнеса ОС Windows.

Эта книга будет полезна администраторам Linux и тем пользователям, которые хотят познакомиться с этой системой поближе. Рассматриваемые вопросы настройки и безопасности пригодятся специалистам, использующим различные ОС, потому что большая часть информации не привязана к определенной системе.

Вы сможете узнать, как хакеры проникают на серверы и защититься от вторжения. Так как некоторые примеры из этой книги могут быть использованы не только для обороны, но и для нападения, я хотел бы предостеречь юных взломщиков. Здоровое любопытство — это хорошо, но помните, что правоохранительные органы не спят и всегда добиваются своего. Если один раз вам повезло со взломом и никто не обратил на это внимания, то в следующий раз вы можете оказаться в руках правосудия.

Часть книги написана с точки зрения хакера и демонстрирует, как они проникают в систему. В надежде на то, что эту информацию не будут использовать для взлома серверов, я старался сделать упор именно на защиту и некоторые вещи оставлял за пределами изложения или просто не договаривал, чтобы не появилось соблазна воспользоваться методами хакеров и нарушить закон. Конечно же, чтобы реализовать мои идеи (довести до логического конца), нужно потратить несколько минут на программирование или на поиск в Интернете. Но несмотря на то, что книга может послужить отправной точкой для хакера, я надеюсь, что этого не произойдет. Помните о законности ваших действий.

Любой объект может быть рассмотрен с разных точек зрения. Простой пример из жизни — нож, являясь столовым предметом, при определенных обстоятельствах становится орудием убийства или средством самообороны. Точно так же и методы хакера, которые будут рассматриваться в этой книге, могут быть восприняты как советы для повседневного ухода за ОС, способы защиты от проникновения и средства взлома системы. Я надеюсь, что вы не будете использовать полученные знания в разрушительных целях, это не украшает человека. Зачем вам нужна "черная" популярность взломщика? Не лучше ли посвятить себя более полезным вещам.

Несмотря на явное стремление Linux поселиться в домашних компьютерах, в этой ОС настройка пока еще слишком сложная и содержит множество параметров, которые большинству пользователей не нужны. Если просто закрыть глаза и оставить все значения по умолчанию, то об истинной безопасности Linux не может быть и речи. Ни одна ОС не будет работать надежно и с максимальной защитой при таких настройках. Производитель не может заранее знать, что нам понадобится, и делает все возможное, чтобы программа работала на любой системе, а для этого приходится включать много дополнительных функций, что делает систему избыточной.

Так уж повелось, что администраторы Linux должны иметь больше опыта и знаний, чем специалисты Windows, и это связано как раз со сложностями настройки. В данной книге я постарался максимально доступно рассказать вам про ОС Linux, и при этом попытался сделать это с позиции хакера.

Почему книга называется "Linux глазами хакера", и что это за глаза? Этот вопрос интересует многих моих читателей. Когда мы берем в руки книгу, то надеемся, что ее внешний вид соответствует внутреннему. В данном случае речь идет о том, какое содержимое будет отвечать этому названию? Для ответа на этот вопрос необходимо четко понимать, кто такой хакер и что он видит в ОС.

Когда меня спрашивают, что я подразумеваю под словом "хакер", я привожу простейший пример: как администратор вы установили и заставили работать ОС, но если удалось настроить ее на максимальную производительность и безопасность, то вы — хакер.

Умения хакера должны быть направлены на то, чтобы создать что-либо лучше других (быстрее, удобнее и безопаснее). Именно такой является сама ОС Linux, сотворенная хакерами для всего мира.

Данная книга рассматривает ОС, начиная с самых основ и до сложных манипуляций системой. Весь излагаемый материал представлен простым и доступным каждому языком. Благодаря этому вам не понадобится дополнительная литература для изучения ОС Linux. Всю информацию можно будет получить из одних рук. Для более глубокого изучения вопроса вам могут

потребоваться только хорошее знание английского языка и чтение документации или файлов HOWTO, которые поставляются с системой Linux.

Главное отличие книги в том, что о безопасности и производительности мы будем говорить не в отдельных заключительных главах, что является большой ошибкой, а по мере необходимости. Когда человек уже приобрел навыки неэффективной работы с системой, то переучиваться будет сложно. Именно поэтому мы будем разбирать последовательно (от азов до сложных вопросов) все аспекты каждой рассматриваемой темы, аккуратно раскладывая полученные знания "по полочкам".

Описание применения и просто администрирование Linux всегда можно найти в Интернете или в документации на ОС, а вот информацию по эффективному использованию найти сложнее, а все имеющиеся сведения являются фрагментарными, и их тяжело сводить в одно целое. А ведь безопасность не любит обрывочных данных. Если упустить хоть одну мелочь, компьютер оказывается уязвимым для взлома.

В качестве дополнительной информации по безопасности компьютера и сетей советую прочитать мою книгу "Компьютер глазами хакера" [3], в которой приводится достаточно много общих сведений по этим вопросам.

Несмотря на то, что данная книга направлена в большей степени на описание безопасности ОС Linux, многие рассматриваемые в ней проблемы могут вам пригодиться и при построении защищенного Linux-сервера. Точно так же книга "Linux глазами хакера" будет полезна и специалистам по безопасности Windows-систем.

В этой книге не рассматриваются вопросы, связанные с вирусами, потому что в настоящее время вирусная активность в ОС Linux минимальна, но это не значит, что опасности не существует. Угроза есть всегда, а защита от вирусов схожа с защитой от троянских программ, которых для Linux достаточно много. Об атаках вирусов и возможности их отражения можно также прочитать в книге "Компьютер глазами хакера" [3].

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

Благодарности

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

Первым делом я хотел бы отметить издательство "БХВ-Петербург", с которым работаю уже несколько лет. Спасибо руководству, редакторам и коррек-

торам, которые работают со мной и помогают сделать книгу такой, какой я ее задумывал. Ведь писать приходится в тяжелых по срокам условиях, но иначе нельзя, т. к. информация устареет раньше, чем книга попадет на прилавок.

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 мысли устремляются далеко в виртуальную реальность.

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

Отдельное спасибо администраторам моего провайдера, которые позволили тестировать некоторые описываемые в данной книги методы на их оборудовании и серверах. Я старался работать аккуратно и ничего не уничтожить. Вроде бы так и получилось ☺.

Так уж выходит, но в написании каждой книги участвуют и животные. Эта работа не стала исключением. Мой новый кот Чикист с 23:00 до 1:00 ночи гуляет по квартире и кричит, я не могу уснуть, а значит, больше времени уделяю работе.

Хочется поблагодарить еще одного кота, который является ассистентом в пакете программ MS Office. Книгу я писал в MS Word, а ОС Linux работала в виртуальной машине, чтобы можно было делать снимки экрана. Если на меня "бросали" ребенка, то кот-ассистент помогал занять моего годовалого сына, выступая в роли няни. Я сажал сына Кирилла рядом, и он спокойно играл с котом на экране монитора, а я мог продолжать работать над книгой. Правда, иногда приходилось спасать кота и монитор, когда сын начинал маленькой ручонкой неуклюже гладить полюбившееся животное.

А самая большая благодарность — вам за то, что купили книгу, и моим постоянным читателям, с которыми я регулярно общаюсь на форуме сайта **www.vr-online.ru**. Последние работы основываются на их вопросах и предложениях. Если у вас появятся какие-то проблемы, то милости прошу на этот форум. Я постараюсь помочь по мере возможности и жду любых комментариев по поводу этой книги. Ваши замечания помогут мне сделать мою работу лучше.

ГЛАВА 1

Введение

Однажды я показывал администратору ОС Windows, как устанавливать и работать с Linux. Сам процесс инсталляции ему понравился, потому что в последних версиях он достаточно прост. Но когда мы установили и решили настроить Samba-сервер, последовала куча вопросов типа: "А зачем настраивать Samba?", "Почему нельзя получить доступ автоматически?" Администраторы Windows-систем ленивы и привыкли, что ОС сама делает за них все, что нужно, но когда их систему взламывают, начинают задавать вопросы: "А почему Microsoft не дала нам нужных инструментов, чтобы запретить определенные действия?".

Если смотреть на ОС Linux с точки зрения пользователя, то после установки системы уже ничего настраивать не надо. Можно сразу же приступать к работе с любыми офисными приложениями и пользовательскими утилитами. Но если речь идет о сетевых и серверных программах, то здесь уже требуются более сложные настройки, и ничего автоматически работать не будет. По умолчанию в системе практически все действия, которые могут привести к нежелательному результату или вторжению по сети, запрещены. Для изменения ограничений нужно настраивать конфигурационные файлы, редактировать которые крайне неудобно, или 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 играться с конфигурационными файлами". Через неделю этот же человек настраивал сервис IIS (Internet Information Services, информационные сервисы Интернета) на новом сервере с ОС Windows 2003. Он ругался теми же словами, потому что эта служба по умолчанию не устанавливается с ОС, и прежде чем начать работать, нужно ее подключить и четко прописать, что должно использоваться, а что нет.

Корпорация Microsoft начинала делать ОС по принципу "лишь бы было удобно", поэтому достаточно было подключить требуемые компоненты. Но теперь Windows становится с каждым годом все сложнее, большинство удобных функций, обеспечивающих защиту, просто отключаются, и при необходимости их приходится открывать. В Linux все было наоборот, эту ОС создавали с точки зрения "лишь бы было безопаснее", а теперь двигаются в сторону наращивания сервисов.

Удобства и безопасность — две стороны одной медали, поэтому производителю приходится чем-то жертвовать.

1.1. Атаки хакеров

Прежде чем знакомится с Linux и ее принципами безопасности, мы должны знать, как хакеры могут проникать в систему. Для того чтобы защитить систему, нужно иметь представление о возможных действиях злоумышленника. Давайте познакомимся с процессом взлома компьютера. Мы должны знать, о чем думают хакеры, чем они дышат и что едят ☺. Только так мы сможем построить непреступную информационную стену для сервера или сети.

Невозможно дать конкретные рецепты взломов. В каждом случае это процесс, который зависит от самой системы и настроек ее безопасности. Чаще всего взлом происходит через ошибки в каких-либо программах, а каждый администратор может использовать различный софт.

Почему количество атак с каждым годом только увеличивается? Раньше вся информация об уязвимостях хранилась на закрытых BBS (Bulletin Board System, электронная доска объявлений) и была доступна только избранным. К этой категории относились и хакеры. Именно они и совершали безнаказанные атаки, потому что уровень просвещенности и опытности таких людей был достаточно высок.

С другой стороны, элита хакерского мира состояла в основном из добропорядочных людей, для которых исследования в области безопасности не являлись целью разрушения.

В настоящее время информация об уязвимостях лежит на каждом углу и является достоянием общественности. Теперь взломом может заниматься кто угодно. Тут же хочется спросить борцов за свободу информации: "Как же так получилось?" Просто чрезмерная свобода в конце концов ведет к разрушению. Есть определенная категория людей, которых хлебом не корми, дай где-нибудь напакостить. Если человек, используя общедоступную информацию, поддастся этой слабости, то он превратится во взломщика.

Злоумышленники при проникновении в систему могут преследовать разные цели.

1. Утечка информации — вскрытие сервера для скачивания каких-либо секретных данных, которые не должны быть доступны широкой общественности. Такие взломы чаще всего направляют против компаний для кражи отчетности, исходных кодов программ, секретной документации и т. д., их выполняют профессиональные хакеры по заказу или для получения собственной выгоды.
2. Нарушение целостности — изменение или уничтожение данных на сервере. Такие действия могут производиться против любых серверов в сетях Интернет/интранет. В качестве взломщиков могут выступать не только профессионалы, но и любители или даже недовольные сотрудники фирм.
3. Отказ от обслуживания — атака на сервер с целью сделать его недосягаемым для остальных участников сети. Этим занимаются, в основном, любители, желая нанести вред.
4. Рабство — получило распространение в последнее время. Сервер захватывается для дальнейшего использования в нападении на другие серверы. Например, для осуществления атаки типа "Отказ от обслуживания" чаще всего нужны значительные ресурсы (мощный процессор и быстрый доступ в сеть), которые отсутствуют на домашнем компьютере. Для осуществления таких атак захватывается какой-либо слабо защищенный сервер в Интернете, обладающий необходимыми ресурсами, и используется в дальнейших взломах.

Атаки могут быть трех видов:

1. Внутренняя — взломщик получил физический доступ к интересующему его компьютеру. Защитить системный блок сервера от злоумышленника не так уж и сложно, потому что можно оградить доступ к серверу сейфом и поставить охрану.
2. Внешняя из глобальной сети — удаленный взлом через сеть. Именно этот вид атаки является самым сложным для защиты. Даже если поставить самый лучший сейф от удаленной атаки (Firewall) и постоянную охрану для наблюдения (программы мониторинга и журналирования), безопасность не может быть гарантированной. Примерами этого являются взломы самых защищаемых серверов в сети (**yahoo.com**, **microsoft.com**, серверы NASA и т. д.).
3. Внешняя из локальной сети — это проникновение, совершенное пользователем вашей сети. Да, хакеры бывают не только в Интернете, соседи по кабинету тоже могут пытаться взломать сервер или ваш компьютер ради шутки или с целью мести.

При построении обороны мы должны понимать, как хакеры атакуют компьютеры своих жертв. Только тогда можно предотвратить нежелательное вторжение и защитить систему. Давайте рассмотрим основные методы нападения, используемые хакером, и способы реализации. Для лучшего понимания процесса будем рассуждать так, как это делает взломщик.

Единственное, чего мы не будем затрагивать, так это вопросы социальной инженерии. Это тема отдельной книги и затрагивать ее не имеет смысла.

1.1.1. Исследования

Допустим, что у вас есть некий сервер, который нужно взломать или протестировать на защищенность от проникновения. С чего нужно начинать? Что сделать в первую очередь? Сразу возникает очень много вопросов и ни одного ответа.

Четкой последовательности действий нет. Взлом — это творческий процесс, а значит, и подходить к нему надо с этой точки зрения. Нет определенных правил и нельзя все подвести под один шаблон. Зато могу дать несколько рекомендаций, которых желательно придерживаться.

Сканирование

Самое первое, с чего начинается взлом или тест системы на уязвимость, — сканирование портов. Для чего? А для того, чтобы узнать, какие сервисы (в Linux это демоны) установлены в системе. Каждый открытый порт — это сервисная программа, установленная на сервере, к которой можно подсоединиться и выполнить определенные действия. Например, на 21 порту висит FTP-сервис. Если вы сможете к нему подключиться, то станет доступной возможность скачивания и закачивания на сервер файлов. Но это только в том случае, если вы будете обладать соответствующими правами.

Для начала нужно просканировать первые 1024 порта. Среди них очень много стандартных сервисов типа FTP, HTTP, Telnet и т. д. Каждый открытый порт — это дверь с замочком для входа на сервер. Чем больше таких дверей, тем больше вероятность, что какой-то засов не выдержит натиска и откроется.

У хорошего администратора открыты только самые необходимые порты. Например, если это Web-сервер, не предоставляющий доступ к почте, то нет смысла включать сервисы почтовых серверов. Должен быть открыть только 80 порт, на котором как раз и работает Web-сервер.

Хороший сканер портов устанавливает не только номер открытого порта, но и определяет установленный на нем сервис. Жаль, что название не настоящее, а только имя возможного сервера. Так, для 80 порта будет показано

"HTTP". Желательно, чтобы сканер умел сохранять результат своей работы в каком-нибудь файле и даже распечатывать. Если этой возможности нет, то придется переписать все вручную и положить на видное место. В дальнейшем вам пригодится каждая строчка этих записей.

После этого можно сканировать порты свыше 1024. Здесь стандартные сервисы встречаются редко. Зачем же тогда сканировать? А вдруг кто-то до вас уже побывал здесь и оставил незапертой дверку или установил на сервер троян. Большинство троянских программ держат открытыми порты свыше 1024, поэтому, если вы администратор и обнаружили такой порт, необходимо сразу насторожиться. Ну а если вы взломщик, то необходимо узнать имя троянской программы, найти для нее клиентскую часть и воспользоваться для управления чужой машиной.

На этом взлом может закончиться, потому что вы уже получили полный доступ к серверу без особых усилий. Жаль, что такое происходит очень редко, чаще всего нужно затратить намного больше усилий.

Лет десять назад сканирование можно было проводить целыми пачками. В настоящее время все больше администраторов устанавливают на свои серверы утилиты, которые выявляют такие попытки сканирования и делают все возможное для предотвращения этого процесса. О том, как защитить свой сервер от сканирования и какие утилиты использовать, мы поговорим в *гл. 12*.

Таким образом, сканирование становится непростой задачей. Сложность заключается в том, что нельзя исследовать порты пачками. Именно поэтому профессионалы предпочитают использовать ручной метод. Для этого достаточно выполнить команду:

```
telnet сервер порт
```

В данном случае с помощью команды `telnet` мы пытаемся подключиться к серверу на указанный порт. Если соединение произошло удачно, то порт открыт, иначе — закрыт. Адрес сервера указывается в качестве первого параметра, а порт — это второй параметр. Если таким образом производить проверку не более 5 портов в час, то большинство программ выявления скана не среагируют, но сам процесс сканирования растянется на недели.

Иногда может помочь утилита nmap, которая позволяет делать сканирование с использованием неполного цикла пакетов. Но современные программы обеспечения безопасности могут обнаружить использование этого метода.

Так как уже на этапе сканирования администратор может занести ваш IP-адрес в список подозреваемых, желательно осуществлять проверку портов не со своего компьютера. Для этого взломщики заводят Web-сайт на сервере, позволяющем встраивать сценарии на языках PHP или Perl, используя бес-

платный хостинг, где во время регистрации не требуют персональных данных, а если запрашивают, то можно ввести неверные данные, потому что их нельзя проверить. После этого к серверу можно подключаться через прокси-сервер и по безопасному соединению управлять собственными сценариями, которые и будут сканировать компьютер жертвы.

Теперь мы в курсе, какие двери у сервера существуют и как это можно определить. Задействованные порты — это всего лишь запертые ворота, и мы пока не знаем, как их открыть. Дальше потребуется больше усилий.

Самым популярным средством сканирования является утилита nmap. Она завоевала сердца хакеров, потому что, во-первых, предоставляет широкие возможности и, во-вторых, не все средства защиты ее определяют. Например, программа антисканирования, установленная на сервере, следит за попытками последовательно или параллельно подключиться на несколько портов. Но nmap может не доводить дело до соединения.

Процесс установки соединения с удаленным компьютером разбивается на несколько этапов. Сначала компьютер посылает пакет с запросом на нужный порт сервера, а тот в ответ должен отправить специальный пакет (не будем вдаваться в подробности протокола TCP, а ограничимся общими понятиями). Только после этого может устанавливаться виртуальное соединение. Сканер nmap может прервать контакт после первого ответа сервера, т. к. уже ясно, что порт открыт и нет смысла продолжать диалог по установке соединения.

Таким образом, программы антисканирования воспринимают такие контакты за ошибки и не сигнализируют администратору о возможной атаке.

Определение ОС

Сканирование — это всего лишь самый начальный этап, который вам еще ничего особенного не сказал. Самое главное перед дальнейшим взломом — определить, какая именно установлена ОС. Желательно знать и версию, но это удается не всегда, да и на первых порах изучения системы можно обойтись без конкретизации.

Как определяется тип ОС? Для этого есть несколько способов.

1. По реализации протокола TCP/IP.

   Различные ОС по-разному реализуют стек протоколов. Программа-клиент (например, nmap) просто анализирует ответы на запросы от сервера и делает вывод об установленной ОС. В основном, это заключение расплывчатое: Windows или Linux. Точную версию таким образом узнать невозможно, потому что в Windows 2000/XP/2003 реализация протокола практически не менялась, и отклики сервера будут одинаковыми. Даже если программа определила, что на сервере установлен Linux, то какой именно

дистрибутив — вам никто не скажет, а ведь уязвимости в них разные. И поэтому такая информация — это только часть необходимых вам данных для взлома сервера.

2. По ответам разных сервисов.

   Допустим, что на сервере жертвы есть анонимный доступ по FTP. Вам нужно всего лишь присоединиться к нему и посмотреть сообщение при входе в систему. По умолчанию в качестве приглашения используется надпись типа "Добро пожаловать на сервер FreeBSD4.0 версия FTP-клиента X.XXX". Если вы такое увидели, то еще рано радоваться, т. к. неизвестно, правда это или нет. Если администратор сервера достаточно опытный, то он, скорей всего, поменяет текст таких сообщений.

   Если надпись приглашения отражает действительность, то администратор — "чайник" со стажем. Опытный администратор всегда изменяет приглашение, заданное по умолчанию. А вот хороший специалист может написать и ложное сообщение. Тогда на сервере с Windows NT 4.0 появится приглашение, например, в Linux. В этом случае злоумышленник безуспешно потратит очень много времени в попытках сломать Windows NT через дыры Linux. Поэтому не очень доверяйте надписям и старайтесь перепроверить любыми другими способами.

Чтобы вас не обманули, обязательно обращайте внимание на используемые на сервере сервисы, например, в Linux не будут крутиться страницы, созданные по технологии ASP. Такие вещи подделывают редко, хотя возможно. Достаточно немного постараться: применить расширение asp для хранения PHP-сценариев и перенаправлять их интерпретатору PHP. Таким образом, хакер увидит, что на сервере работают файлы asp, но реально это будут PHP-сценарии.

Следовательно, задача защищающейся стороны — как можно сильнее запутать ситуацию. Большинство неопытных хакеров верят первым впечатлениям и потратят очень много времени на бесполезные попытки проникновения. Таким образом, вы сделаете взлом слишком дорогим занятием.

Задача хакера — распутать цепочки и четко определить систему, которую он взламывает. Без этого производить в дальнейшем какие-либо действия будет сложно, потому что хакер даже не будет знать, какие команды ему доступны после вторжения на чужую территорию и какие исполняемые файлы можно подбрасывать на сервер.

Для определения ОС хакеры любят использовать утилиту nmap. Основные возможности программы направлены на сканирование портов, но если запустить программу с параметром -o, то она попытается определить тип операционной системы. Конечно же, существует вероятность ошибки, но возможна и правильная работа.

Используем скрипты

Итак, теперь вы знаете, какая на сервере установлена ОС, какие порты открыты и какие именно серверы висят на этих портах. Вся эта информация должна быть у вас записана в удобном для восприятия виде: в файле или хотя бы на бумаге. Главное, чтобы комфортно было работать.

На этом можно остановить исследования. У вас есть достаточно информации для простейшего взлома с помощью дыр в ОС и сервисах, установленных на сервере. Просто посещайте регулярно **www.securityfocus.com**. Именно здесь нужно искать информацию о новых уязвимостях. Уже давно известно, что большая часть серверов (по разным источникам от 70 до 90 %) просто не латаются. Поэтому проверяйте все найденные ошибки на жертве, возможно, что-то и сработает.

Если сервер в данный момент близок к совершенству, то придется ждать появления новых дыр и сплоитов (программа, позволяющая использовать уязвимость) к установленным на сервере сервисам. Как только увидите что-нибудь интересное, сразу скачайте сплоит (или напишите свой) и воспользуйтесь им, пока администратор сервера не залатал очередную уязвимость.

1.1.2. Взлом WWW-сервера

При взломе WWW-сервера есть свои особенности. Если на нем выполняются CGI/PHP или иные скрипты, то взлом проводится совершенно по-другому. Для начала нужно просканировать сервер на наличие уязвимых CGI-скриптов. Вы не поверите, но опять же по исследованиям различных компаний в Интернете работает большое количество "дырявых" скриптов. Это связано с тем, что при разработке сайтов изначально вносятся ошибки. Начинающие программисты очень редко проверяют входящие параметры в надежде, что пользователь не будет изменять код странички или адрес URL, где серверу передаются необходимые данные для выполнения каких-либо действий.

Ошибку с параметрами имела одна из знаменитых систем управления сайтом — PHP-nuke. Это набор скриптов, позволяющих создать форум, чат, новостную ленту и управлять содержимым сайта. Все параметры в скриптах передаются через строку URL браузера, и просчет содержался в параметре ID. Разработчики предполагали, что в нем будет передаваться число, но не проверяли это. Хакер, знающий структуру базы данных (а это не сложно, потому что исходные коды PHP-nuke доступны), легко мог поместить SQL-запрос к базе данных сервера в параметр ID и получить пароли всех зарегистрированных на сайте пользователей. Конечно, клиенты будут зашифрованы, но для расшифровки не надо много усилий, и это мы рассмотрим чуть позже *(см. разд. 14.10)*.

Проблема усложняется тем, что некоторые языки (например, Perl) изначально не были предназначены для использования в сети Интернет. Из-за этого в них

существуют опасные функции для манипулирования системой, и если программист неосторожно применил их в своих модулях, то злоумышленник может воспользоваться такой неосмотрительностью.

Потенциально опасные функции есть практически везде, только в разных пропорциях. Единственный более или менее защищенный язык — Java, но он очень сильно тормозит систему и требует много ресурсов, из-за чего его не охотно используют Web-мастера. Но даже этот язык в неуклюжих руках может превратиться в большие ворота для хакеров с надписью: "Добро пожаловать!".

Но самая большая уязвимость — неграмотный программист. Из-за нехватки специалистов в этой области программированием стали заниматься все, кому не лень. Многие самоучки даже не пытаются задуматься о безопасности, а взломщикам это только на руку.

Итак, ваша первостепенная задача — запастись парочкой хороших CGI-сканеров. Какой лучше? Ответ однозначный — ВСЕ. Даже самый дрянной сканер может найти брешь, о которой неизвестно даже самому лучшему хакеру. А главное, что по закону подлости именно она окажется доступной на сервере. Помимо этого, нужно посещать все тот же сайт **www.securityfocus.com**, где регулярно выкладываются описания уязвимостей различных пакетов программ для Web-сайтов.

1.1.3. Серп и молот

Там, где не удалось взломать сервер с помощью умения и знаний, всегда можно воспользоваться чисто русским методом "Серпа и молота". Это не значит, что серп нужно приставлять к горлу администратора, а молотком стучать по его голове. Просто всегда остается в запасе тупой подбор паролей.

Давайте снова обратимся к статистике. Все исследовательские конторы пришли к одному и тому же выводу, что большинство начинающих выбирают в качестве пароля имена своих любимых собачек, кошечек, даты рождения или номера телефонов. Хорошо подобранный словарь может сломать практически любую систему, т. к. всегда найдутся неопытные пользователи с такими паролями. Самое страшное, если у этих "чайников" будут достаточно большие права.

Вы до сих пор еще не верите мне? Давайте вспомним знаменитейшего "червя Морриса", который проникал в систему, взламывая ее по словарю. Собственный лексикон червя был достаточно маленький и состоял менее чем из ста слов. Помимо этого, при переборе использовались термины из словаря, установленного в системе. Там их было тоже не так уж много. И вот благодаря такому примитивному алгоритму червь смог поразить громаднейшее число

компьютеров и серверов. Это был один из самых массовых взломов!!! Да, случай давний, но средний профессионализм пользователей не растет, т. к. среди них много опытных, но достаточно и начинающих.

1.1.4. Локальная сеть

Взлом в локальной сети может быть проще по многим причинам:

- компьютеры подключены по скоростному соединению от 10 Мбит и выше;
- есть возможность прослушивать трафик других компьютеров в сети;
- можно создавать подставные серверы;
- очень редко используются сетевые экраны, потому что их ставят в основном перед выходом в Интернет.

Рассмотрим различные варианты взломов, которые получили наибольшее распространение.

Прослушивание трафика

В локальной сети есть свои особенности. Соединения могут осуществляться различными способами. От выбранного типа топологии зависит используемый вид кабеля, разъем и используемое оборудование. При подключении по коаксиальному кабелю могут использоваться две схемы: все компьютеры объединяются напрямую в одну общую шину или кольцо. Во втором случае крайние компьютеры тоже соединены между собой, и когда они обмениваются данными, все пакеты проходят через сетевую карту соседнего компьютера.

Если используется такая топология, то весь трафик обязательно проходит через все компьютеры сети. Почему же вы его не видите? Просто ОС и сетевой адаптер сговорились и не показывают чужой трафик. Но если очень сильно захотеть, то, воспользовавшись программой-сниффером, можно и просмотреть все данные, проходящие мимо сетевой карточки, даже если они предназначены не вам.

При подключении к Интернету с помощью снифера можно увидеть только свой трафик. Чтобы позаимствовать чужую информацию, нужно сначала взломать сервер провайдера, а туда уже поставить снифер и смотреть трафик всех клиентов. Это слишком сложно, поэтому способ через сниферы используют чаще всего в локальных сетях.

Соединение по коаксиальному кабелю встречается все реже, потому что оно ненадежно и позволяет передавать данные максимум на скорости 10 Мбит/с.

При объединении компьютеров через хаб (Hub) или коммутатор (Swich) используется топология "Звезда". В этом случае компьютеры с помощью витой

пары получают одну общую точку. Если в центре стоит устройство типа Hub, то пакеты, пришедшие с одного компьютера, копируются на все узлы, подключенные к этому хабу. В случае с коммутатором пакеты будет видеть только получатель, потому что Swich имеет встроенные возможности маршрутизации, которые реализуются в основном на уровне MAC-адреса. В локальной сети, даже если вы отправляете данные на IP-адрес, применяется физический адрес компьютера. При работе через Интернет всегда используется IP-адресация, а на этом уровне работают далеко не все коммутаторы. Для того чтобы пакеты проходили по правильному пути, нужны уже более интеллектуальные устройства — маршрутизаторы. Они также передают набор данных только на определенную машину или другому маршрутизатору, который знает, где находится компьютер получателя.

Таким образом, при использовании коммутаторов в локальной сети и из-за маршрутизации в Интернете прослушивание становится более сложным, и для выполнения этой задачи снифер должен находиться на самом коммутаторе или маршрутизаторе.

Работать с пакетами достаточно сложно, потому что в них содержится трудная для восприятия информация. Большой кусок данных разбивается на несколько пакетов, и вы будете видеть их по отдельности.

Сейчас в сети можно скачать громадное количество сниферов и дополнений к ним. Всевозможные версии "заточены" под разные нужды, и тут необходимо выбирать именно из этих соображений. Если вы ищете непосредственно пароли, то вам требуется снифер, который умеет выделять данные о регистрации из общего трафика сети. В принципе, это не так сложно, если учесть, что все пароли и любая информация пересылаются в Интернет в открытом виде как текст, если не используется SSL-протокол (Secure Socket Layer, протокол защищенных сокетов).

Преимущество от использования сниферов при взломе в том, что они никак не влияют на атакуемую машину, а значит, вычислить его очень сложно. В большинстве случаев даже невозможно узнать, что ваш трафик кем-то прослушивается в поисках паролей.

Подставной адрес

Очень часто серверы ограничивают доступ с помощью правил для сетевого экрана. Но блокировать абсолютно все обращения к любым портам не всегда удобно. Например, доступ к управляющим программам можно сохранить для определенного IP-адреса, с которого работает администратор. Каждый, кто попытается с другого адреса войти в запрещенную область, будет остановлен сетевым экраном.

На первый взгляд, защита безупречна. Но существует такой метод атаки, как спуфинг, который подразумевает подделку IP-адреса авторизованного поль-

зователя и вход в штурмуемый сервер. Старые сетевые экраны (да и дешевые современные) не могут определить фальшивый адрес в пакетах. Хороший Firewall может направить ping-команду компьютеру, чтобы убедиться, что он включен, и именно он пытается запросить нужные ресурсы.

Фиктивный сервер

В локальной сети намного проще производить атаку через подставные серверы или сервисы. Например, одна из знаменитых атак через некорректные ARP-записи может быть воспроизведена именно в локальной сети.

Когда вы обращаетесь к какому-либо компьютеру по IP-адресу, сначала определяется его MAC-адрес, а потом уже на него отсылается сообщение. Как определить MAC-адрес, когда нам неизвестно, какой сетевой интерфейс установлен, а мы знаем только IP? Для этого используется протокол ARP (Address Resolution Protocol, протокол разрешения адресов), который рассылает широковещательный запрос всем компьютерам сети и выясняет, где находится экземпляр с указанным IP-адресом. В этом пакете заполнен только IP-адрес, а вместо искомого MAC-адреса указано значение FFFFFFFFFFFFh. Если в сети есть компьютер с запрошенным IP, то в ответном пакете будет указан MAC-адрес. Работа ARP-протокола происходит незаметно для пользователя. В противном случае ответ может прислать маршрутизатор, который сообщит свой MAC-адрес. Тогда компьютер будет обмениваться данными с ним, а тот уже в свою очередь будет пересылать пакеты дальше в сеть или другому маршрутизатору, пока они не достигнут получателя.

А что если ответит не тот компьютер, а другой, с иным IP-адресом? Ведь в локальной сети передача осуществляется по MAC-адресу, поэтому пакет получит тот компьютер, который откликнется, независимо от его IP. Получается, что задача хакера вычислить ARP-запрос и ответить на него вместо реального адресата. Таким образом можно перехватить соединение.

Допустим, что компьютер запросил соединение с сервером. Если мы ответим на него и сэмулируем запрос на ввод параметров для входа в сервер, то пароль будет перехвачен. Сложность такого метода в том, что вручную его реализовать практически невозможно. Для этого нужно писать соответствующую программу, а тут без знания программирования не обойтись.

1.1.5. Троян

Использование троянских программ — самый глупый и ненадежный в отношении администраторов сетей способ, но для простых пользователей подойдет, потому что им проще подбросить серверную часть программы. Хотя среди администраторов встречаются непрофессионалы, но на такие шутки уже мало кто попадается. Но кто сказал, что в сети существуют только они? Есть

еще куча простых пользователей с большими привилегиями и доверчивой душой. Вот именно их и надо троянить.

Троянская программа состоит из двух частей — клиент и сервер. Сервер нужно подбросить на компьютер жертвы и заставить его запустить файл. Чаще всего троянская программа прописывается в автозагрузку и стартует вместе с ОС и при этом незаметна в системе. После этого вы подключаетесь к серверной части с помощью клиента и выполняете заложенные в программу действия, например, перезагрузка компьютера, воровство паролей и т. д.

Как забрасывать троянскую программу? Самый распространенный способ — почтовый ящик. Просто даете исполняемому файлу серверной части какое-нибудь привлекательное имя и отправляете сообщение жертве. В тексте письма должны быть мягкие, но заманчивые призывы запустить прикрепленный файл. Это то же самое, что и распространение вирусов, письма с которыми мы видим каждый день в своих ящиках. Если пользователь запустит серверную часть, то считайте, что вы стали царем на его компьютере. Теперь вам будет доступно все, что может для вас сделать боевой конь.

Если троянская программа направлена на воровство паролей, то после заражения она может незаметно для пользователя выслать письмо с файлом паролей на определенный E-mail-адрес. Профи легко находят такие адреса (с помощью отладки приложения), но на этом все останавливается. Профессиональные хакеры не глупы и для троянских программ регистрируют почтовые адреса на бесплатных сервисах, при этом указывается ложная информация о владельце. Злоумышленник заводит почтовый ящик или проверяет его на предмет писем с паролями только через анонимный прокси-сервер, и узнать реальный IP-адрес человека становится очень сложно.

Трояны получили большое распространение из-за того, что вычислить автора при соблюдении простых правил анонимности нелегко. При этом использование самих программ стало примитивным занятием.

Опасность, которую таят в себе троянские программы, подтверждается и тем, что большинство антивирусных программ стали сканировать не только на наличие вирусов, но и троянов. Например, антивирусные программы идентифицируют Back Orifice, как вирус Win32.BO.

1.1.6. Denial of Service

Самая глупая атака, которую могли придумать хакеры, — это отказ от обслуживания (DoS). Заключается она в том, чтобы заставить сервер не отвечать на запросы. Как это можно сделать? Очень часто такого результата добиваются с помощью зацикливания работы. Например, если сервер не проверяет корректность входящих пакетов, то хакер может сделать такой запрос, кото-

рый будет обрабатываться вечно, а на работу с остальными соединениями не хватит процессорного времени, тогда клиенты получат отказ от обслуживания.

Атака DoS может производиться двумя способами: через ошибку в программе или перегрузку канала/мощности атакуемой машины.

Первый способ требует знания уязвимостей на сервере. Рассмотрим, как происходит отказ от обслуживания через переполнение буфера (чаще всего используемая ошибка). Допустим, что вы должны передать на сервер строку "HELLO". Для этого в серверной части выделяется память для хранения 5 символов. Структура программы может выглядеть примерно следующим образом:

```
Код программы
Буфер для хранения 5 символов
Код программы
```

Предположим, пользователь отправит не пять, а сто символов. Если при приеме информации программа не проверит размер блока, то при записи данных в буфер они выйдут за его пределы и запишутся поверх кода. Это значит, что программа будет испорчена и не сможет выполнять каких-либо действий, и, скорее всего, произойдет зависание. В результате сервер не будет отвечать на запросы клиента, т. е. совершится классическая атака Denial of Service через переполнение буфера.

Таким образом, компьютер не взломали, информация осталась нетронутой, но сервер перестал быть доступным по сети. В локальной сети такую атаку вообще несложно произвести. Для этого достаточно свой IP-адрес поменять на адрес атакуемой машины, и произойдет конфликт. В лучшем случае недоступной станет только штурмуемая машина, а в худшем — обе машины не смогут работать.

Для перегрузки ресурсов атакуемой машины вообще не надо ничего знать, потому что это война, в которой побеждает тот, кто сильнее. Ресурсы любого компьютера ограничены. Например, Web-сервер для связи с клиентами может организовывать только определенное количество виртуальных каналов. Если их создать больше, то сервер становится недоступным. Для совершения такой акции достаточно написать программу на любом языке программирования, бесконечно открывающую соединения. Рано или поздно предел будет превышен, и сервер не сможет работать с клиентами.

Если нет программных ограничений на ресурсы, то сервер будет обрабатывать столько подключений, сколько сможет. В таком случае атака может производиться на канал связи или на сервер. Выбор цели зависит от того, что слабее. Например, если на канале в 100 Мбит стоит компьютер с процессором Pentium 100 МГц, то намного проще убить машину, чем перегрузить

данными канал связи. Ну а если это достаточно мощный сервер, который может выполнять миллионы запросов в секунду, но находится на канале в 64 Кбит, то легче загрузить канал.

Как происходит загрузка канала? Допустим, что вы находитесь в чате, и кто-то вам нагрубил. Вы узнаете его IP-адрес и выясняете, что обидчик работает на простом соединении Dial-up через модем в 56 Кбит/с. Даже если у вас такое же соединение, можно без проблем перегрузить канал обидчику. Для этого направляем на его IP-адрес бесконечное количество ping-запросов с большим размером пакета. Компьютер жертвы должен будет отвечать на них. Если пакетов много, то мощности канала хватит только на то, чтобы принимать и отвечать на эхо-запросы, и обидчик уже не сможет нормально работать в сети. Если у вас канал такой же, то и ваше соединение будет занято исключительно приемом-отсылкой больших пакетов. Таким образом, мы можем загрузить канал жертвы, но при условии, что характеристики нашего канала сопоставимы с атакуемым компьютером. Если у вас скорость соединения медленней, то удастся загрузить только часть канала, равную пропускной способности вашего соединения. Остальная часть останется свободной, и жертва сможет использовать ее. С другой стороны, связь будет заторможена, и хотя бы чего-то мы добьемся. Это того стоит? Если да, то можете приступать.

В случае атаки на сервер и его процессор наш канал может быть намного слабее, главное — правильно определить слабое звено. Допустим, что сервер предоставляет услугу скачивания и хранения файлов. Чтобы перегрузить канал такого сервера, нужно запросить одновременное получение нескольких очень больших файлов. Скорость связи резко упадет, и сервер может даже перестать отвечать на запросы остальных клиентов, при этом загрузка процессора сервера может быть далека от максимальной. Если неправильно определить слабое звено (в данном случае это сеть) и нарастить мощность сервера, то производительность все равно не увеличится.

Для загрузки процессора тоже не требуется слишком большой канал. Нужно только подобрать запрос, который будет выполняться очень долго. Допустим, что вы решили произвести атаку на сервер, позволяющий переводить на другой язык указанные страницы любого сайта. Находим Web-страницу с большим количеством текста (например, книгу или документацию RFC — Request for Comments, рабочее предложение) и посылаем множество запросов на ее перевод. Мало того, что объем большой, и для скачивания серверу нужно использовать свой канал, так еще и перевод — довольно трудоемкий процесс. Достаточно в течение 1 секунды отправить 100 запросов на перевод громадной книги, чтобы сервер перегрузился. А если используется блокировка мно-

гократного перевода одного и того же, то нужно подыскать несколько больших книг.

Атака отказа от обслуживания отражается достаточно просто. Серверное программное обеспечение должно контролировать и ограничивать количество запросов с одного IP-адреса. Но это все теоретически, и такие проверки оградят только от начинающих хакеров. Опытному взломщику не составит труда подделать IP-адрес и засыпать сервер пакетами, в которых в качестве отправителя указан поддельный адрес.

Для сервера еще хуже, если взлом идет по TCP/IP, потому что этот протокол требует установки соединения. Если хакер пошлет очень большое количество запросов на открытие соединения с разными IP-адресами, то сервер разошлет на эти адреса подтверждения и будет дожидаться дальнейших действий. Но т. к. реально с этих адресов не было запроса, то и остановка будет лишенной смысла. Таким образом, заполнив буфер очереди на входящие соединения, сервер становится недоступным до момента подключения несуществующих компьютеров (TimeOut для этой операции может быть до 5 секунд). За это время хакер может забросать буфер новыми запросами и продлить бессмысленное ожидание сервера.

Distributed Denial of Service

С помощью DoS достаточно сложно вывести из обслуживания такие серверы, как **www.microsoft.com** или **www.yahoo.com**, потому что здесь достаточно широкие каналы и сверхмощные серверы. Получить такие же ресурсы просто невозможно. Но как показывает практика, хакеры находят выходы из любых ситуаций. Для получения такой мощности используются распределенные атаки DoS (Distributed Denial of Service).

Мало кто из пользователей добровольно отдаст мощность своего компьютера для проведения распределенной атаки на крупные серверы. Чтобы решить эту проблему, хакеры пишут вирусы, которые без разрешения занимаются захватом. Так вирус Mydoom C искал в сети компьютеры, зараженные вирусами Mydoom версий A и B, и использовал их для атаки на серверы корпорации Microsoft. Благо этот вирус не смог захватить достаточного количества машин, и мощности не хватило для проведения полноценного налета. Администрация Microsoft утверждала, что серверы функционировали в штатном режиме, но некоторые все же смогли заметить замедление в работе и задержки в получении ответов на запросы.

От распределенной атаки защититься очень сложно, потому что множество реально работающих компьютеров шлют свои запросы на один сервер. В этом случае трудно определить, что это идут ложные запросы с целью вывести систему из рабочего состояния.

1.1.7. Взлом паролей

Когда взломщик пытается проникнуть в систему, то он чаще всего использует один из следующих способов:

- если на атакуемом сервере уже есть аккаунт (пусть и гостевой), то можно попытаться поднять его права;
- получить учетную запись конкретного пользователя;
- добыть файл паролей и воспользоваться чужими учетными записями.

Даже если взломщик повышает свои права в системе, он все равно стремится обрести доступ к файлу с паролями, потому что это позволит добраться до учетной записи `root` (для Unix-систем) и получить полные права на систему. Но пароли зашифрованы, и в лучшем случае можно будет увидеть hash-суммы, которые являются результатом необратимого шифрования пароля.

Когда администратор заводит нового пользователя в системе, то его пароль чаще всего шифруется с помощью алгоритма MD5, т. е. не подлежит дешифровке. В результате получается hash-сумма, которая и сохраняется в файле паролей. Когда пользователь вводит пароль, то он также шифруется, и результат сравнивается с hash-суммой, хранящейся в файле. Если значения совпали, то пароль введен верно. О хранении паролей в Linux мы еще поговорим в *разд. 4.3*.

Так как обратное преобразование невозможно, то, вроде бы, и подобрать пароль для hash-суммы нельзя. Но это только на первый взгляд. Для подбора существует много программ, например, John the Ripper (**http://www.openwall.com/john/**) или Password Pro (**http://www.insidepro.com/**).

Почему эти утилиты так свободно лежат в Интернете, раз они позволяют злоумышленнику воровать пароли? Любая программа может иметь как положительные, так и отрицательные стороны. Что делать, если вы забыли пароль администратора, или администратор уволился и не сказал вам его? Переустанавливать систему? Это долго и может грозить потерей данных. Намного проще снять жесткий диск и подключить к другому компьютеру (или просто загрузиться с дискеты, умеющей читать вашу файловую систему), потом взять файл паролей и восстановить утерянную информацию.

1.1.8. Итог

Каждый взломщик в своем арсенале имеет множество методов взлома, количество которых зависит от опыта. Чем искушеннее хакер, тем больше вариантов он собирает и отрабатывает на сервере. Определив ОС и запущенные на сервере сервисы, взломщик начинает последовательно использовать известные ему приемы атаки.

Конечно же, подбор паролей доступен всем хакерам, но к нему прибегают в последнюю очередь, потому что он может отнять слишком много времени и не принести результата. Даже перебор всех паролей окажется неработоспособным, если сервер настроен на выявление таких попыток, и администратор отреагирует на них нужным образом — добавит в настройки сетевого экрана запрет на подключение с IP-адреса, который использовал хакер для подбора. Все остальные действия злоумышленника станут бесполезными, пока он не сменит свой адрес.

Этот обзор хакерских атак не претендует на полноту, но я постарался дать все необходимые начальные сведения. В то же время я воздержался от конкретных рецептов, потому что это может быть воспринято, как призыв к действию, а я не ставлю своей целью увеличить количество хакеров. Моя задача — показать, как хакер видит и использует компьютер. Это поможет вам больше узнать о своем помощнике и сделать собственную жизнь безопаснее.

В основном мы рассматривали теорию. Для реализации на практике всего вышесказанного нужны специализированные программы, и для определенных задач их придется писать самостоятельно.

Вы обязаны хорошо понимать теорию взлома для того, чтобы ведать, от чего защищаться. Не зная этого, вы не сможете построить полноценную оборону, позволяющую отразить даже простые атаки хакера.

Для защиты собственного дома от мелких хулиганов достаточно хорошего замка и сигнализации, а против грабителей и убийц надо ставить решетки на окнах, железные двери и колючую проволоку на заборе.

Интернет слишком большой, и в нем живут хакеры различной квалификации, использующие разные способы взлома. Вы должны располагать информацией о возможных нападениях. Сложно предугадать, каким именно методом воспользуется злоумышленник. Нужно быть готовым ко всему и уметь отразить атаку самостоятельно.

1.2. Что такое Linux?

Linux — это свободная операционная система, исходные коды которой открыты для всеобщего просмотра и даже внесения изменений.

Основа ядра ОС была создана в 1991 году студентом Хельсинского университета (University of Helsinki) по имени Линус Торвальдс (Linus Torvalds). Он написал костяк, функционально схожий с Unix-системами, и выложил его для всеобщего просмотра с просьбой помогать ему в улучшении и наращивании возможностей новой ОС. Откликнулось достаточно много людей, и работа закипела.

Хакеры из различных стран присоединились к этому проекту на общественных началах и начали создавать самую скандальную ОС. А буза вокруг Linux возникает чуть ли не каждый день, потому что ОС получила большое распространение и является абсолютно бесплатной. Некоторые производители программного обеспечения считают этот проект перспективным, другие (например, Microsoft) — периодически превращают во врага.

Официальная версия ядра ОС под номером 1.0 была выпущена в 1994 году, т. е. через три года после первых "слухов" о Linux. Такая скорость разработки была достигнута благодаря большому количеству профессионалов, которые согласились развивать интересную задумку Линуса.

ОС Linux — это многопользовательская и многозадачная система, которая позволяет работать с компьютером сразу нескольким пользователям и выполнять одновременно разные задачи.

Почему именно эта ОС получила такое признание, ведь были и есть другие открытые проекты, а по реализации даже лучше Linux? Я связываю эту популярность с тем, что Linux создавался хакерами и для хакеров. Очень приятно, когда ты работаешь в операционной системе, в которой есть частичка тебя. Любой пользователь может вносить в исходный код системы любые изменения и не бояться преследования со стороны закона.

Изначально популярность среди администраторов росла благодаря тому, что эта ОС поддерживала основные стандарты Unix, к которым относятся POSIX, System V и BSD. При этом система была написана для дешевой (по сравнению с дорогостоящими серверами Sun и IBM) платформы x86 и обладала всеми необходимыми возможностями. Таким образом, многие фирмы смогли оптимизировать свои расходы на организацию инфраструктуры информационных технологий (ИТ) за счет перевода некоторых серверных задач на бесплатный продукт — Linux.

Среди первых задач, которые стали доверять Linux, — организация Web-сервера, и с ней ОС справляется великолепно. Трудно точно оценить, какой процент пользователей сейчас использует Linux, но большинство статистических анализов показывает, что на Linux совместно с сервером Apache приходится большая часть.

На данный момент в Linux можно сделать практически все. Для этой ОС написано уже множество продуктов, которые распространяются бесплатно и позволяют решать всевозможные задачи. Компьютеры с установленной Linux используют в различных областях науки, экономики и техники, в том числе и при создании специальных эффектов для кино.

Не менее важным фактором популярности стала демократичность ОС. Вас не ограничивают в возможностях и не заставляют следовать определенным предпочтениям разработчика. В комплект поставки ОС включается по не-

сколько программ одинакового назначения, например, несколько браузеров или офисных программ. В Windows такое невозможно. Мы, наверное, никогда не увидим в одном дистрибутиве браузеры Internet Explorer, Mozilla и Opera (самостоятельные коммерческие продукты). В Linux конкуренция действительно свободная, и никто не запрещает использовать сторонние разработки и не борется с этим.

1.3. Открытый исходный код – это безопасно?

Бытует мнение, что программы с открытым исходным кодом надежнее и безопаснее, чем коммерческие.

Сторонники этого утверждения считают, что такую систему исследуют много людей разными способами и тем самым выявляют все возможные погрешности. ОС Windows XP показывает достаточно высокую надежность и безопасность, хотя является коммерческим продуктом. А самое главное, что ошибки исправляются своевременно, доступны для свободного скачивания и легки в установке.

Да, искать ошибки на уровне кода совместно с тестированием готового продукта намного проще и эффективнее, но результат далек от идеала. Несмотря на массовое тестирование в Linux достаточно часто находят ляпсусы. А если посмотреть, какая армия пользователей обследовала последние версии Windows, то можно было бы подумать, что она станет безупречной. Тестирование — это одно, а применение в "боевых условиях" демонстрирует совершенно непредсказуемые результаты.

Открытость в отношении Linux имеет одно преимущество — отличное соотношение цены и качества. Возможность бесплатно установить ОС позволяет сэкономить большие деньги. Но затраты появятся в плане поддержки, которая для Linux стоит достаточно дорого, поэтому со своевременными обновлениями могут возникнуть проблемы. К тому же администрирование Linux требует больших навыков и умений, чем Windows. Отсутствуют мастера, которые облегчат жизнь. Необходимо знать команды Linux и уметь ими пользоваться без подсказки. Именно поэтому эта ОС до сих пор не поселилась в наших домах.

Почему же Linux так сложна? Ответ прост. Производительность и удобство — несовместимые вещи. В Windows все предельно ясно, но для выполнения какой-либо операции может понадобиться множество щелчков мыши и просмотр нескольких диалоговых окон, что отнимает драгоценное время. В Linux нужно запустить консоль и выполнить нужную директиву. Проблема только в том, что надо помнить множество команд.

ОС Windows везде, где только можно, использует визуальное представление и графический интерфейс. В Linux же графические утилиты слишком просты и зачастую не обладают достаточными возможностями, но это поправимо, и сейчас появляется все больше оконных утилит, упрощающих процесс настройки. Пройдет какое-то время, и Linux станет тривиальной в применении и при этом сохранит всю мощь и скорость использования командной строки.

Так как настройка Linux — достаточно сложная процедура, требующая высокой квалификации, очень часто именно из-за неправильно установленных параметров эта система попадает под огонь хакеров. Любая система (Windows, Linux или Mac OS X) с настройками по умолчанию далека от идеала. Часто безопасностью жертвуют для обеспечения производительности или удобства. В некоторых программах включаются сервисные функции, которые облегчают работу администратора (например, отладка в интерпретаторе PHP), но и упрощают взлом со стороны хакера. Именно поэтому безопасность системы напрямую зависит только от человека, который ее обслуживает.

Наша задача не просто научиться работать с ОС Linux, а делать это эффективно, т. е. суметь настроить ее на максимальную производительность и безопасность. Именно такую цель мы и поставим перед собой.

И все же, безопасность Linux выше, чем Windows, и это не связано с открытостью или закрытостью исходного кода. В Linux многие вопросы решены лучше, чем в Windows, например, работа с памятью, определенная область которой выделяется при запуске программы. Выйти за ее пределы невозможно (только в крайних случаях, для обмена данными с другими приложениями). А в Windows каждая программа может получить доступ к любому участку памяти, вплоть до системного. Это грозит тем, что программа по ошибке может изменить чужую область памяти и даже разрушить систему.

Начиная с Windows 2000, подсистема работы с оперативной памятью улучшилась, но она все равно несовершенна. Например, ОС Linux после завершения программы сама может очистить память, потому что точно знает, где, сколько и под какие нужды было ее выделено. В Windows это реализовать сложнее, поэтому можно только надеяться на совершенствование кода.

1.4. Ядро

Ядро — это сердце ОС, в котором реализовано управление физическими и программными ресурсами компьютера. Помимо этого оно позволяет получить доступ к различному железу. Например, ранние версии ядра обеспечивали работу только двух USB-устройств: клавиатура и мышь. Начиная с версии 2.4, встроена поддержка USB-видеокамер, принтеров и других устройств.

Номер версии ядра Linux состоит из трех чисел:

- первое (старший номер) — указывает на значительные изменения в ядре;
- второе (младший номер) — характеризует появление небольших изменений. По нему можно определить, является ядро проверенным или предназначено для тестирования и нет уверенности, что оно не содержит ошибок. Если число четное, то ядро прошло тщательное тестирование. В противном случае установка данной версии не гарантирует стабильной работы;
- третье — номер очередного рабочего релиза (сборка). В некоторых случаях это число опускают. Например, мы в этой главе уже говорили о ветке 2.4, и в данном случае не указана именно сборка.

Вы должны самостоятельно обновлять ядро или помогать в тестировании нестабильных версий. Новые версии ядра можно скачать по адресу **www.kernel.org** или с сайта производителя вашего дистрибутива.

Обновление ядра позволяет не только получить новые возможности по работе с железом, повысить производительность системы, но и исправить некоторые ошибки, которые есть всегда и везде. Самое главное, что обновление ядра в Linux не влечет за собой переконфигурирования всей ОС, как это происходит в некоторых других системах. Я видел компьютеры, которые были установлены еще несколько лет назад и не перенастраивались с тех пор, а только обновлялось ядро и программное обеспечение. Такое бывает редко, потому что, как правило, периодически необходимо обновлять железо, наращивая мощности, потому что запросы программ и пользователей растут не по дням, а по часам.

1.5. Дистрибутивы

На данный момент существует множество различных дистрибутивов Linux, но несмотря на это легко проглядывается схожесть между ними, т. к. большинство имеет общие корни. Например, многие дистрибутивы построены на основе Red Hat Linux. Компании-производители вносят некоторые коррективы в инсталляцию (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 почти везде используется KDE или/и GNOME, а при отсутствии в поставке их всегда можно установить. Таким образом, в не зависимости от основного дистрибутива у всех будет одинаковый графический интерфейс.

В данной книге мы будем рассматривать один из клонов самого распространенного дистрибутива Red Hat (по некоторым данным составляет около 50 % рынка Linux). В качестве примера я выбрал ASPLinux, как самый популярный в России. Если вы будете пользоваться другим дистрибутивом, то особой разницы не заметите (только в графической части во время установки).

Разнообразие дистрибутивов является самым слабым звеном ОС Linux. Когда вы начнете работать с ОС, то увидите, что большая часть операций не стандартизирована (можно расценивать как следствие открытости кода). Получается как в поговорке "Кто в лес, кто по дрова". Это серьезная проблема, которая усложняет восприятие.

В этом смысле ОС Windows более унифицирована и проще для обучения. Хотя в последнее время и здесь наблюдается отступление от установленных канонов. Так внешний вид программ стал совершенно непредсказуем. Меню и панели в Office 2000/XP/2003 постоянно изменяются (только успевай привыкать к ним!). В Linux, несмотря на отсутствие стандартов, элементы интерфейса везде одинаковы.

Дистрибутивы Linux являются условно бесплатными. Их также нужно приобретать в коробочном варианте, но их лицензионное соглашение намного мягче, чем у коммерческих ОС. Например, купив одну коробку с Linux, вы можете устанавливать ее на любое количество рабочих станций.

Цена одной копии Linux намного ниже, чем Windows, и при этом в дистрибутив входит громадное количество офисных программ, интернет-утилит, графических редакторов и т. д. Таким образом, установив полную версию, ваш компьютер готов сразу решать большинство производственных и домашних задач.

В ОС Windows графический редактор (Paint), текстовый процессор (WordPad) и другие программы слишком примитивны, и для нормальной работы нужно потратить тысячи долларов. Поэтому реальная стоимость рабочего места на базе ОС Windows намного выше цены дистрибутива Linux.

При таком сравнении ОС Linux окажется победителем. Но у Windows бесплатная поддержка, а для получения полноценной помощи в Linux нужен доступ к сети Red Hat, который стоит достаточно дорого. Поэтому расходы на поддержку могут сравнять стоимости владения этими операционными системами. Именно поэтому я не буду вас убеждать, что Linux лучше, потому что бесплатная, это не совсем верно. Но мы увидим, что ОС Linux более гибкая, надежная и, зачастую, выигрывает по производительности. Именно эти качества являются наиболее существенными, и я покажу, что все они присущи Linux.

Итак, давайте рассмотрим основные дистрибутивы, которые вы можете встретить на рынке. Помните, что Linux — это всего лишь ядро, а большин-

ство программ, сервисов и графическая оболочка принадлежат сторонним разработчикам. Какой именно продукт будет включен в состав ОС, определяется производителем дистрибутива.

Ваш выбор должен зависеть от того, что именно вы хотите получить от системы, но и это не является обязательным, потому что любой дистрибутив можно "нарастить" дополнительными пакетами программ.

1.5.1. Red Hat Linux

Данный дистрибутив считается классическим и является законодателем моды в развитии ОС, потому что именно в этой фирме работает основатель Linux — Линус Торвальдс. Для получения этого дистрибутива вы можете купить коробочный вариант или скачать версию бесплатно с сайта **www.redhat.com**. Помимо этого, Red Hat ведет разработку ОС Linux в двух направлениях: для серверных решений и для клиентских компьютеров. Второй вариант начал все больше обретать дружественный интерфейс и способен решить любые домашние задачи.

Установка этого дистрибутива уже давно стала простой и удобной. В следующей главе мы будем рассматривать, как инсталлируется клон ОС Red Hat, и вы поймете, что это не так уж и сложно.

Все дистрибутивы Linux всегда ругают за сложность установки ядра и программ, которые чаще всего поставляются в исходных кодах и требуют компиляции. Компания Red Hat упростила этот процесс с помощью разработки пакетов RPM (Redhat Package Manager).

Многие почитатели Linux надеются, что благодаря Red Hat их любимая ОС станет доступной всем и сможет победить конкурентов.

Если вы выбираете себе дистрибутив для сервера, то я настоятельно рекомендую обратить внимание на этого производителя или его клонов, потому что Red Hat заботится о безопасности системы и старается исправлять ошибки с максимально возможной скоростью.

1.5.2. Slackware

Мое знакомство с Linux начиналось именно с дистрибутива Slackware (**www.slackware.com**). Это один из самых старых и сложных дистрибутивов для домашних пользователей. До сих пор нет удобной программы установки, и большинство действий приходится делать в текстовом режиме. Конечно же, вы можете добавить к этому дистрибутиву KDE или GNOME (графические оболочки), а также другие пакеты, облегчающие работу, но установку проще не сделаешь.

Если вы ни разу не работали с Linux, то я бы не рекомендовал начинать знакомство с этого дистрибутива. Лучше выбрать что-нибудь попроще.

1.5.3. SuSE Linux

Мне приходилось работать с разными программами от немецких производителей, но их юзабилити не просто хромало, такие программы — это безногие калеки с детства. Но разработка от SuSE (**www.suse.de**) опровергает мое мнение. Этот дистрибутив отличается симпатичным интерфейсом и отличной поддержкой оборудования, потому что содержит громадную базу драйверов.

Честно сказать, я бы удивился, если бы кто-либо умудрился испортить KDE или GNOME, ведь внешний вид ОС зависит от этих оболочек. В данном случае фирма SuSE смогла не испортить эту красоту своими картинками и логотипами.

Но нельзя сказать, что программисты SuSE вообще ничего не сделали. Они добавили в дистрибутив набор утилит под названием YaST, которые значительно упрощают администрирование. Простота — это хорошо, но максимальных преимуществ можно добиться только при прямом конфигурировании файлов.

Я бы посоветовал SuSE только любителям и для использования на клиентских компьютерах.

1.5.4. 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 этим дистрибутивом 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.

Существует еще множество дистрибутивов, и их возможности варьируются от больших и мощных систем, включающих все необходимое, до маленьких дистрибутивов, загружающихся с дискеты и работающих на очень старых компьютерах.

Основная задача книги — создание безопасной и быстрой системы — усложняется из-за большого количества дистрибутивов. Описать особенности каждого из них в одной книге очень сложно, да и не имеет смысла, потому что способы реализации защиты могут отличаться от дистрибутива к дистрибутиву и даже от версии к версии ядра. Слишком динамично развивается Linux.

На этом вводную часть закончим и перейдем непосредственно к установке ОС Linux, чтобы начать знакомиться с системой на практике и увидеть все своими глазами.

ГЛАВА 2

Установка и начальная настройка Linux

Установка всегда была самой сложной процедурой у 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 были примеры, связанные со случайной потерей данных.

Во время инсталляции любая ОС должна определить установленное оборудование и подготовить все необходимое для его нормальной работы. Еще семь лет назад перечень поддерживаемых 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 чтение такого списка займет дни, потому что все крупные игроки компьютерного мира начали считаться с пингвином (животное, которое ассоциируют с Linux). Определение оборудования теперь происходит безошибочно и, чаще всего, не требует дополнительного вмешательства со стороны пользователя.

В настоящее время вся инсталляция происходит практически автоматически и сравнима по сложности с установкой других ОС. Именно поэтому корпорация Microsoft начинает бояться Linux и ее продвижения в бездну домашних компьютеров. Теперь уже любой, даже начинающий пользователь справится с установкой. И все же мы бегло рассмотрим этот процесс и остановимся на наиболее интересных моментах.

Если вы уже имеете опыт установки Linux, я все же рекомендую вам прочитать эту главу, потому что некоторые детали могут оказаться интересными и полезными. Основные принципы безопасности и производительности закла-

дываются уже на этапе установки, и впоследствии мы будем только следовать им и расширять наши познания.

2.1. Подготовка к установке

Какой дистрибутив устанавливать? Я не могу ничего советовать, потому что выбор всегда остается за вами. Отдайте предпочтение тому, который удовлетворяет вашим потребностям и может решить поставленные задачи. В *разд. 1.5* мы рассмотрели наиболее популярные на данный момент дистрибутивы и их основные отличия, что облегчит вам принятие правильного решения.

Единственное замечание, которое я должен сделать, — устанавливайте самый свежий дистрибутив, включающий последнюю версию ядра и приложений. Мы говорили и будем еще не раз повторять, что во всех программах есть ошибки, просто в новой версии о них никто еще не знает ☺. Кроме того, надо помнить, что программные средства необходимо своевременно обновлять. Если воспользоваться старым дистрибутивом, то объем обновлений может оказаться слишком велик. Не лучше ли установить сразу все новое и максимально быстро запустить сервер в эксплуатацию.

Мастера установок для разных дистрибутивов могут отличаться, но все они, как правило, имеют схожие окна и даже последовательность выполняемых действий зачастую одинакова.

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 во время установки просто отведите под нее все доступное пространство.

Если у вас уже установлена Windows, и вы хотите, чтобы на компьютере было сразу две ОС, то придется сделать несколько телодвижений. Для этого нужно пустое пространство на диске. Нет, это не свободное место на логическом диске **C:**, а пустота на винчестере. В последних инсталляционных версиях есть возможность изменять размер диска прямо в программе установки. В противном случае придется воспользоваться программой Partition Magic (**http://www.powerquest.com/partitionmagic**).

Запустите Partition Magic. Главное окно программы вы можете увидеть на рис. 2.1. Слева находится дерево, в котором отображаются все жесткие диски, установленные в системе. В данном случае имеется только один диск, внутри которого создан основной раздел **C:**, занимающий полное пространство. В правой части окна есть графическое представление соотношения занятого (отображается более темным цветом) и свободного пространства, в рамках которого и можно освобождать диск.

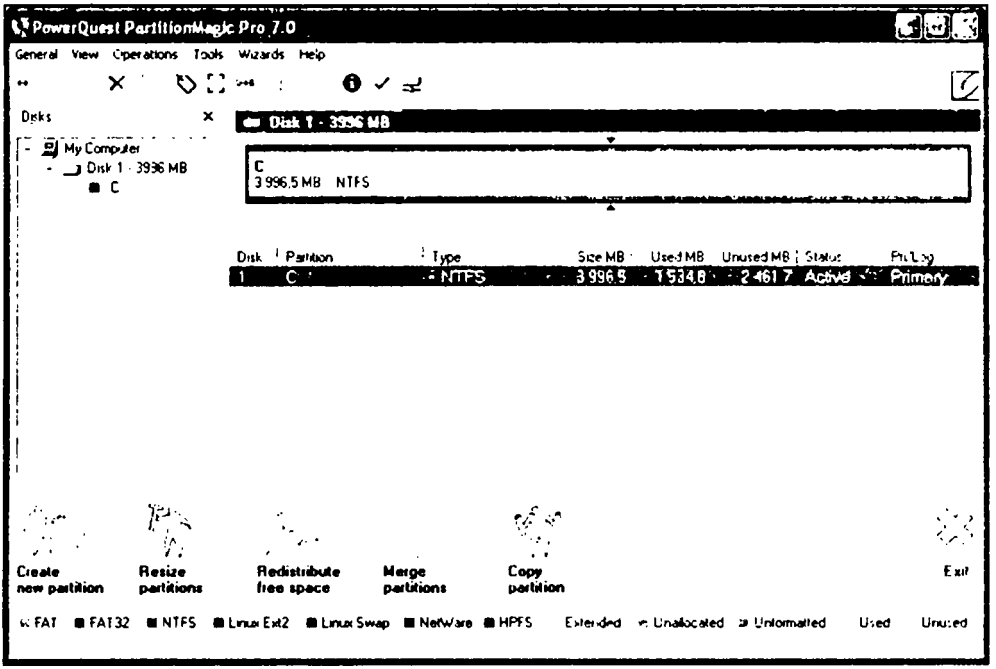

Рис. 2.1. Главное окно программы Partition Magic

Наша задача уменьшить логический диск C:, чтобы на высвободившемся месте ставить Linux.

Выделите диск C: в дереве или в графическом представлении, нажмите кнопку **Resize partitions**, и перед вами появится окно, в котором можно указать новый размер выбранного диска. Для Linux желательно освободить не менее 4 Гбайт, а лучше и больше. Если ваш диск был размером 20 Гбайт, то укажите в качестве нового значения 16, и у вас освободятся требуемые 4 Гбайта. Чтобы применить трансформацию, нужно нажать кнопку **Exit**. Программа запросит подтверждение на внесение изменений и может предупредить о необходимости перезагрузиться в DOS-режим. Соглашайтесь. Все дальнейшие действия произойдут автоматически, и в следующий раз Windows начнет работу с уже уменьшенным диском.

2.2. Начало установки

Итак, свободное место на диске у нас уже есть. Теперь можно приступить к запуску инсталлятора. Для этого вставьте диск в CD-ROM и перезагрузите компьютер. Если в BIOS (Basic Input/Output System, базовая система ввода/вывода) не поставлена загрузка с CD-ROM, то самое время сделать это,

тогда при старте компьютера автоматически запустится процедура установки. Вначале на экране побегут строчки тестирования компьютера, установленных жестких дисков, CD-ROM, мыши, видеокарты, монитора и т. д. На первый взгляд весь текст, который вы будете видеть, весьма жуткий и отпугивающий. Но это только кажется. ОС Windows также тестирует систему, просто ничего не показывает пользователю. Если слишком страшно, то закройте глаза, через минуту все пройдет.

Опросив оборудование, программа установки Linux перейдет в графический режим, и перед вами появится приятное окно выбора языка. Мы пока разговариваем на русском, поэтому укажем его и нажмем кнопку **Далее**.

На втором этапе нужно выбрать мышь (рис. 2.2). Программа установки не сильно доверяет своему исследованию внешних устройств, и правильно делает. У меня почти всегда определяется двухкнопочная мышь PS/2. Третью кнопку Linux даже не пытается искать. В принципе это нормальная ситуация, потому что в Linux почти везде используются две кнопки, но я всегда ставлю третью.

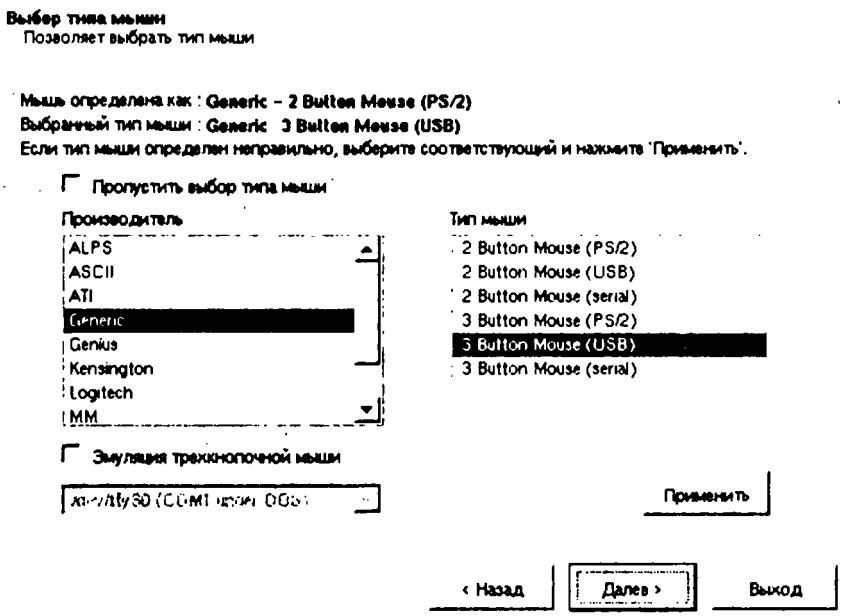

Рис. 2.2. Подтверждение правильности определения мыши

Окно подтверждения мыши содержит два взаимосвязанных списка: производители и соответствующие ему известные типы устройства. Чтобы не забы-

вать себе голову, остановитесь на производителе **Generic** (общие настройки, которые будут работать с устройствами любого производителя), а в правом списке выберите строку с желаемым количеством кнопок и нужным портом. Подключение для мыши указано в скобках и на данный момент может быть трех типов:

- PS/2 — современный порт для подключения устройств ввода (клавиатура, мышь);
- USB — универсальный интерфейс, используется для различных устройств, встречается все чаще и чаще;
- Serial — в народе известен еще и как COM-порт, встречается в старых мышках. Сейчас такой антиквариат найти уже сложно.

На следующем шаге вам предлагается выбрать один из трех вариантов установки:

1. Быстрая — конфигурация определена производителем. Таким образом вы сократите количество вопросов, на которые надо ответить во время установки, но результат будет далек от оптимального или безопасного.
2. Выборочная — позволяет самостоятельно указать желаемые компоненты. Наиболее предпочтительный вариант для серверов и компьютеров, которые будут подключены к сети.
3. Обновление существующей версии — используется при уже установленной Linux. Это позволит потратить меньше времени на повторную настройку.

Затем следует указать, где находятся файлы установки. Так как мы загрузились с инсталляционного диска, то, конечно же, нужно выбрать пункт **CD-ROM**. Современные версии Linux можно легко установить и с жесткого диска, и по сети.

Следующий этап — выбор диска, куда нужно устанавливать ОС. Это отдельная песня с бубнами и флейтой, которая может вызвать затруднения, поэтому я постараюсь остановиться на этом шаге более подробно.

2.3. Разбивка диска

В ASPLinux предусмотрено три варианта использования дискового пространства для размещения ОС:

1. Весь диск. В этом случае все существующие разделы будут уничтожены, а значит, вся информация будет потеряна. Этот вариант удобен, если вы устанавливаете единственную ОС на новый компьютер. Программа установки сама выберет, сколько места и для чего отвести.

2. Свободное место. Если на компьютере уже есть установленная ОС, и вы освобождали пустое пространство с помощью Partition magic, то выбирайте этот пункт. Программа установки создаст диски для Linux, исходя из свободного пространства на жестком диске.
3. Дополнительно. Дает возможность самостоятельно выбрать параметры создаваемых дисков. Этот вариант наиболее сложный, но позволяет добиться максимально эффективных и безопасных результатов.

2.3.1. Именование дисков

В Linux диски нумеруются не так, как мы привыкли в Windows. Здесь нет диска **A:**, **C:** и т. д. Все диски имеют имена **/dev/hdaX**, где X — это номер диска. Поясню, в каком диапазоне должны быть номера. Для первого жесткого диска назначается цифра 1 (**/dev/hda1**), для второго — 2 и т. д. до 4, потому что всего в компьютере может быть установлено 4 жестких диска (хотя в современных серверах используются намного большие массивы). Затем идут логические диски: **/dev/hda5**, **/dev/hda6** и т. д.

На первый взгляд это сложно, но давайте рассмотрим пример, и все встанет на свои места. Допустим, что у вас есть два жестких диска. Первый из них разбит на два раздела, назовем их A — основной и B — дополнительный. На втором диске у нас будет один большой раздел C. Теперь посмотрим, какие имена назначит этим дискам ОС Linux:

A — **/dev/hda1**;

C — **/dev/hda2**;

B — **/dev/hda5**.

Основной раздел первого диска получил номер 1. Цифра 2 присвоена основному разделу второго диска. Дополнительный раздел первого диска имеет номер 5. Если бы на втором диске были дополнительные разделы, то они получили номера, начиная с 6. Как видите, значения 3 и 4 остались свободными, потому что у нас только два жестких диска.

2.3.2. Файловые системы

Теперь поговорим о файловых системах, с которыми работает Linux. Эта ОС поддерживает множество систем, в том числе и Windows-файловые системы FAT, FAT32 и NTFS, но при установке ОС Linux желательно выбрать родную систему Ext2, Ext3 или ReiserFS (это название часто сокращают до Reiser). Последняя является новинкой и наиболее предпочтительна, потому что включает журналирование, которое делает систему более устойчивой и позволяет быстро восстанавливать ее после сбоев.

Рассмотрим, как работают файловые системы, чтобы вы смогли выбрать оптимальный вариант. В файловой системе Ext2 данные сначала кэшируются и только потом записываются на диск, за счет этого достигается высокая производительность. Но если возникнут проблемы с питанием или произойдет аварийный выход из системы, то компьютер может не успеть сохранить данные. При следующем входе ОС обнаружит нарушение целостности жесткого диска и запустится программа сканирования диска fsck (как scandisk в Windows), которая восстановит его работоспособность, но воссоздать утерянные данные уже не удастся. Сканирование занимает много времени, и это может сказаться на скорости возобновления работы сервера. Будьте готовы к тому, что следующая загрузка будет происходить дольше обычного.

В файловой системе ReiserFS также выполняется запись с предварительным кэшированием, а после этого проверка целостности данных, и если все записано верно, то кэш очищается. В противном случае ОС при запуске быстро найдет проблемные места с помощью созданного журнала и с минимальными потерями времени восстановит работоспособность диска.

У файловой системы ReiserFS есть еще одно преимущество. Данные на жесткий диск записываются блочно. Допустим, что блок занимает 1 Кбайт. Если, например, в FAT32 записать файл размером 0,1 Кбайт, то блок будет уже занят, но в нем останется 90 % пустого пространства, которое нельзя использовать, т. е. происходит утечка памяти на жестком диске. Таким образом, на нем будет храниться немного меньше информации, чем вы ожидали. Файловая система RaiserFS позволяет лучше заполнять блоки.

Утечку наглядно можно увидеть, если в ОС Windows открыть **Свойства** файла (рис. 2.3). Обратите внимание, что в окне есть два параметра **Size** (Размер) и **Size on disk** (Размер на диске). Величина файла 4,95 Кбайт, а на диске он занимает целых 8 Кбайт. Арифметика простая, понятно, что один кластер на диске равен 4 Кбайтам. Размер файла больше этого значения, поэтому ОС пришлось выделить два блока, и второй заполнен менее чем на 25 %. Остальное дисковое пространство пропало и не может использоваться.

Если на диск поместить 1000 файлов по 100 байт при размере кластера 4 Кбайта, то каждый из них запишется в свой блок. При этом на диске будет израсходовано 400 Кбайт вместо положенных 100. Потери пространства составят 75 %.

Файловая система ReiserFS позволяет записывать в один блок несколько файлов, если их размер менее 100 байт. Таким образом, на диске будет меньше дыр и утечки памяти.

Файловая система Ext3 — новое поколение журналируемых систем и работает по аналогии с ReiserFS. На данный момент она является системой по умолчанию в большинстве современных дистрибутивов Linux. Трудно срав-

нить по производительности ReiserFS и Ext3, но с точки зрения надежности советую использовать вторую. Стоит согласиться с мнением разработчиков.

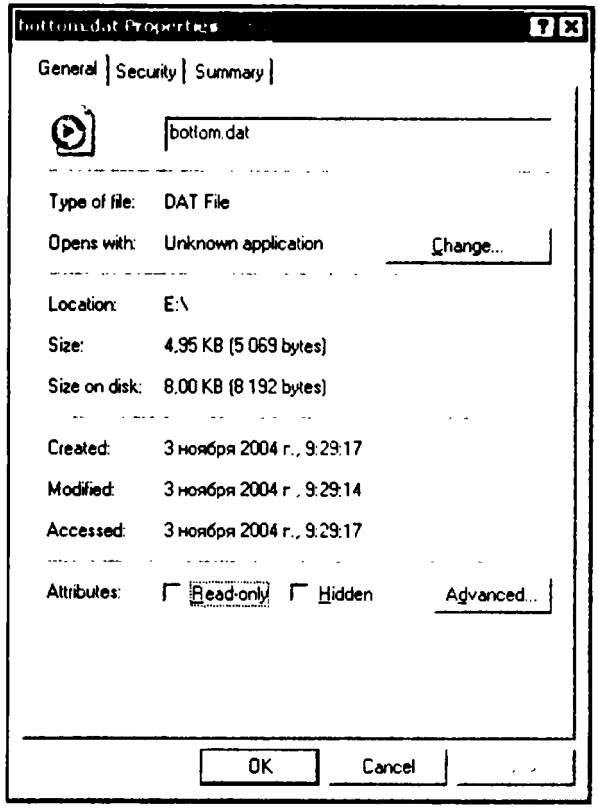

Рис. 2.3. Окно свойств файла

2.3.3. Ручное создание разделов

Если вы настраиваете сервер, а не домашний компьютер, то я настоятельно рекомендую создать разделы вручную. По умолчанию программа установки создаст только два раздела — основной и для файла подкачки (будет использоваться при нехватке оперативной памяти), это неэффективно и даже небезопасно. Итак, выбираем параметр **Дополнительно**, чтобы самостоятельно создать разделы. Перед вами откроется окно, как на рис. 2.4.

Вверху окна находится выпадающий список (**Диск**), где можно указать диск, на котором нужно создать раздел. В данном случае выбран диск **/dev/hda**. Чуть ниже расположена таблица созданных на этом диске разделов. Как видно из рис. 2.4, пока разделов нет и все пространство пустое.

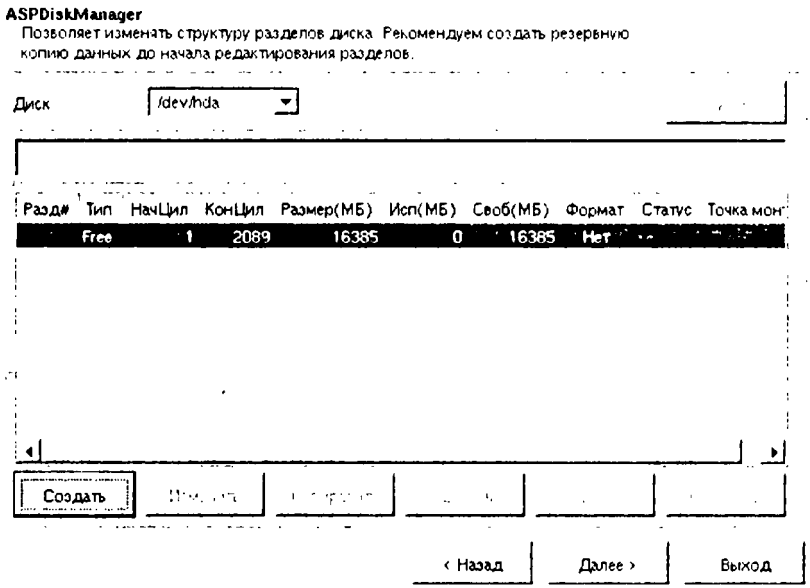

Рис. 2.4. Окно управления разделами

С уже существующими разделами можно делать следующие операции: изменять параметры, копировать, двигать и удалять. Для этого щелкните мышью на интересующем разделе и нажмите соответствующую кнопку внизу окна.

Для создания диска нужно выделить строку пустого пространства и нажать кнопку **Создать**. Перед вами появится окно, как на рис. 2.5.

В выпадающем списке **Тип файловой системы** можно выбрать один из предложенных видов. Некоторые из этих систем (Ext3, Ext2, Reiser) мы уже рассмотрели в *разд. 2.3.2*. Познакомимся с остальными:

- XFS — устаревшая система, поддерживающая диски до 2 Гбайт и максимальный размер файла в 64 Мбайта;
- Swap — используется только для разделов подкачки, о которых поговорим чуть ниже;
- RAID — в данной книге мы эту тему затрагивать не будем.

Чуть ниже расположена полоска, с помощью которой можно указать размер будущего диска простым перетаскиванием ее мышью. Для любителей работать руками есть возможность задать размер явным образом в поле **Размер** (вводится число в мегабайтах).

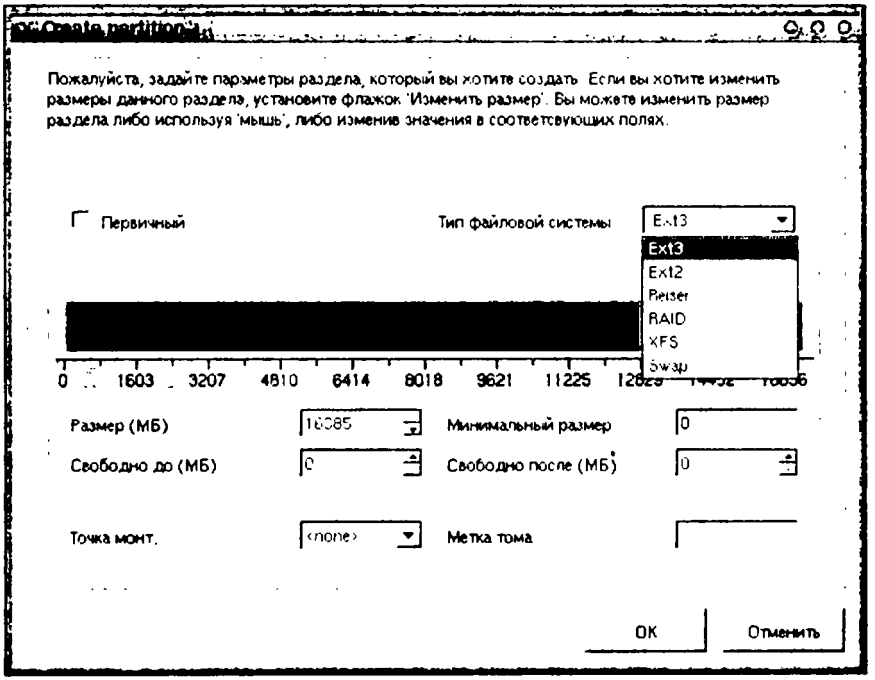

Рис. 2.5. Окно создания раздела

В выпадающем списке **Точка монт.** можно указать точку (папку), которая будет храниться в данном разделе.

В табл. 2.1 перечислены разделы, которые можно создавать, и их назначение. В различных дистрибутивах некоторые из этих разделов могут отсутствовать.

Таблица 2.1. *Разделы, которые можно создать*

Раздел	Описание
/	Основной раздел. Если в Windows при указании пути к файлу сначала указывается имя диска, а потом папки внутри него, то здесь началом является слэш
/bin	Основные исполняемые файлы системы
/boot	Файлы, необходимые для загрузки системы
/dev	Представления для подключенных устройств
/etc	Конфигурационные системные файлы
/home	Пользовательские папки и файлы
/lib	Библиотеки ядра ОС

Таблица 2.1 (окончание)

Раздел	Описание
/opt	Дополнительные программные пакеты
/proc	Для монтирования виртуальной файловой системы
/sbin	Исполняемые файлы главного пользователя (root)
/tmp	Временные файлы
/usr	Системные файлы
/var	Журналы, буферные или заблокированные файлы
Swap	Раздел подкачки

Обратите внимание, что все имена разделов, кроме первого и последнего, начинаются со слэша. Это неслучайно, потому что первый и последний разделы являются обязательными, а все остальные представлены как вложенные в основной раздел папки.

Swap — это всегда отдельный раздел со своей файловой системой и не может быть представлен в виде папки корневого диска. Размер этого раздела должен быть не меньше объема доступной оперативной памяти. Если у вас достаточно места на жестком диске, то я советую сделать **Swap** в 3 раза больше, чем установленное ОЗУ, потому что память может быть расширена, а вот файл подкачки потом изменить будет проблематично.

Для работы Linux нужно создать как минимум два раздела: основной (обозначается как /) и подкачки (**Swap**). Первый раздел может иметь любой тип файловой системы, кроме Swap, а второй как раз наоборот, должен быть типа Swap. В основном разделе будут располагаться все файлы, а второй — будет использоваться для расширения оперативной памяти. Если остальные разделы, перечисленные в табл. 2.1, не созданы, то они будут представлены в основном разделе в виде папок.

На первый взгляд, все сложно, особенно если вы до этого работали только с ОС Windows. Но поверьте мне, что возможность создания многочисленных папок ОС в виде отдельных разделов является действительно мощной возможностью. Два раздела 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, будет использоваться при нехватке памяти;
- **/home** — на втором диске для хранения пользовательских файлов.

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

Если вы настраиваете сервер, то на отдельные жесткие диски можно вынести папки **/home** и **/var**. Логические диски/разделы в этом случае не дадут желаемого результата.

Какими должны быть разделы? **Swap** нужно задавать в зависимости от установленной памяти, и об этом мы уже говорили чуть раньше. Если разделы **/var** и **/home** вынесены на отдельные диски, то для корневого каталога будет достаточно 4 Гбайт, но можно сделать 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 будем возвращаться. 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 с разделом **/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.

Ориентируйтесь на свои технические средства. При необходимости разделы **/var** и **/home** действительно можно разместить на одном, но самом большем жестком диске. Только вот под раздел **/home** нужно отдавать все оставшееся пространство, потому что здесь пользователи будут хранить свои данные (достигают большого объема!), и если пожадничать, то скоро начнутся возмущения со стороны пользователей о нереальности сохранения на сервере результатов очередной игры. Если возможность есть, то выделите каждому из разделов максимально большой диск и забудете про проблемы.

Для тестовой системы можно выбрать простейший вариант с двумя разделами (корневой и подкачки) и продолжить установку.

Следующий шаг после создания всех разделов — их форматирование. Некоторые особо умные дистрибутивы выполняют эту операцию без дополнительных вопросов.

2.4. Выбор пакетов для установки

Следующий этап заключается в определении компонентов, которые надо устанавливать. Это достаточно интересный момент, и именно здесь обычно допускают первую и самую страшную ошибку — указывают все. Да, названия и назначение многих пакетов непонятны и незнакомы большей части пользователей, поэтому начинающие не могут четко определить список того, что им необходимо. Но это не значит, что нужно устанавливать все подряд.

У меня на отладочной системе действительно стоит Linux в полной комплектации со всем, что только можно. На ней я тестирую новые программы и проверяю работоспособность отдельных модулей. Но в "боевых" системах не устанавливаю ничего лишнего.

Любой дистрибутив Linux включает в себя невероятное множество программ, особенно серверных. Тут и Web-сервер, и FTP, и многое другое. Установив все, мы делаем свой компьютер "проходным двором", особенно если все это активизируется при старте системы. Загрузка компьютера замедлится и станет сравнима с запуском Windows XP на Pentium 100.

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. Если хотя бы в одном демоне (серверная программа, которая обрабатывает запросы клиента) найдется погрешность, то любой хакер сможет проникнуть в вашу систему и делать в ней все, что вздумается.

Для рабочей системы я всегда устанавливаю абсолютно голую ОС, а затем добавляю только то, что нужно, особенно это касается серверных программ. Нарастить компоненты можно в любой момент, а вот отказаться от существующих порой бывает очень сложно.

Итак, Linux предлагает нам выбрать один из следующих варнантов установки:

- **Типовая** — включает приложения, которые разработчик посчитал часто используемыми;
- **Разработка** — будут установлены основные пакеты и все необходимые компоненты для разработки приложений, компиляции ядра ОС Linux и т. д.;
- **Офис** — типовые пакеты плюс офисные приложения;
- **Сервер** — только ОС и демоны;
- **Пользовательский** — вы сами можете сформировать нужный список.

Никогда не выбирайте серверную установку. В этом случае на компьютере отсутствуют клиентские приложения, но зато будут работать в фоновом ре-

жиме всевозможные программы-демоны. Это как раз самый опасный вариант. Если офисные программы не открывают портов, не работают с сетью (т. е. не могут нанести ущерб серверу извне), а также не загружаются в память при старте и не влияют на производительность, то серверные приложения тормозят систему, и каждый лишний демон — это удар по безопасности.

Я рекомендую остановить свой выбор на пункте **Разработка**, чтобы у нас были все необходимые возможности для написания программ и работы с документами, но при этом в компьютере не устанавливались серверные приложения. После определения типа инсталляции укажите пункт **Выборочно** внизу окна. Это позволит вам выбрать дополнительно требуемые пакеты.

На следующем этапе мы увидим список всех компонентов, которые могут быть установлены. По умолчанию в списке нет выделенных серверных программ (и не надо). Но если заранее известно о необходимости какого-либо сервера, то можно его отметить, чтобы он автоматически установился. Допустим, вы знаете, что вам обязательно понадобится Web-сервер. Для этого чаще всего используют Apache. Найдите в дереве компонентов пункт **Web Server**, раскройте его и поставьте галочку напротив **apache** (основные файлы сервера) и **apacheconf** (программа настройки). Если вы будете писать программы на языке PHP, то здесь же можно выбрать все необходимые компоненты для такой разработки.

Не пожалейте времени и пройдитесь по всем имеющимся в списке пакетам. Выберите только самое необходимое, впоследствии в любой момент мы сможем расширить возможности. Помните, что уже на этапе установки мы закладываем фундамент будущей производительности и безопасности системы.

Не устанавливайте ничего лишнего. Если вы не пользуетесь какой-либо программой, то, конечно же, не будете следить за ее обновлениями и не станете заниматься исправлениями ошибок. Для повышения своих привилегий злоумышленник сможет воспользоваться и такой программой. Таким образом, вы откроете нараспашку дверь хакеру.

Когда вы выберете все необходимые пакеты и нажмете кнопку **Далее**, начнется их непосредственное копирование на жесткий диск. Это займет достаточно много времени, поэтому можно приготовить чашечку кофе или даже успеть посмотреть какой-нибудь фильм.

Пока идет установка, поговорим еще немного об этом процессе, чтобы к моменту настройки системы вы были во всеоружии. Допустим, что в вашей сети должны работать три сервера: Web-, FTP- и сервер новостей. Все эти функции может выполнять один компьютер, но безопасность в этом случае будет далека от идеала. Я всегда разношу каждую задачу на отдельные компьютеры и вам советую не экономить на железе и делать то же самое.

Каждый запущенный демон — это потенциальная дыра. Как мы уже знаем, в них неизменно существуют погрешности, и не всегда администраторы узнают о них первыми. Допустим, что ошибка найдена в сервере Apache. В последнее время это происходит достаточно редко, потому что программа уже очень хорошо отлажена, но представим эту ситуацию. Ошибка может быть не в самом сервере Apache, а в обслуживаемом им Web-сайте или в интерпретаторе PHP/Perl. В любом случае хакер может воспользоваться этой брешью и с легкостью получит доступ к FTP-серверу и скачает все секретные данные. Если на данном компьютере будет работать только Web-сервер, то доступ через FTP к конфиденциальным данным будет проблематичен. Максимум, что может сделать хакер, — дефейс (замена главной страницы) или уничтожение сайта. Но это восстановить проще, чем воссоздавать все данные с FTP- или сервера новостей.

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 еще поговорим в *гл. 4*, но уже сейчас вы должны знать, что для каждой системы должен быть свой код доступа.

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 поставляются в исходниках. Для того чтобы ими воспользоваться, злоумышленник закачивает такой модуль на сервер и выполняет программу. Чтобы компиляция стала невозможной, я рекомендую не устанавливать библиотеки разработчика и компилятор gcc.

В ОС Linux очень редко используются инсталляторы программ, поэтому все настройки производятся во время компиляции исходного кода. Если gcc будет недоступнен, то у взломщика возникнут проблемы с выполнением зловредного кода.

Конечно же, опытный хакер соберет программу из исходных кодов на своем компьютере и после этого закачает на взломанный сервер, но у начинающего злоумышленника отсутствие компилятора может вызвать затруднения. А ведь каждая проблема для взломщика — это победа специалиста по безопасности.

Если вы только осваиваетесь в мире Linux, то я могу посоветовать установить linuxconf — это пакет программ, упрощающий администрирование и включенный в поставку. Во время изучения Linux вы увидите, что очень много настроек приходится делать в конфигурационных файлах вручную. Это не обязательно, т. к. сейчас существует большое количество программ, упрощающих настройку и имеющих визуальный интерфейс.

Если вы можете обойтись без linuxconf, то лучше настраивать файлы вручную. Программы с визуальным интерфейсом нередко вносят в конфигурацию далеко небезопасные параметры или устанавливают высокие права доступа для сервисов. Поэтому после работы linuxconf не мешает проверить внесенные программой изменения, а для этого вы должны отлично знать структуру и содержание конфигурационных файлов.

2.5. Завершение установки

После завершения копирования файлов на диск нужно разобраться еще с несколькими настройками. Во-первых, система должна знать, как вы ее будете загружать. Если Windows без разговоров прописывает загрузку в MBR (Master Boot Record, основная загрузочная запись), то Linux позволяет выбрать любой другой диск или вообще его не прописывать. В этом случае читать Linux можно будет только с дискеты или некоторыми другими мудреными способами.

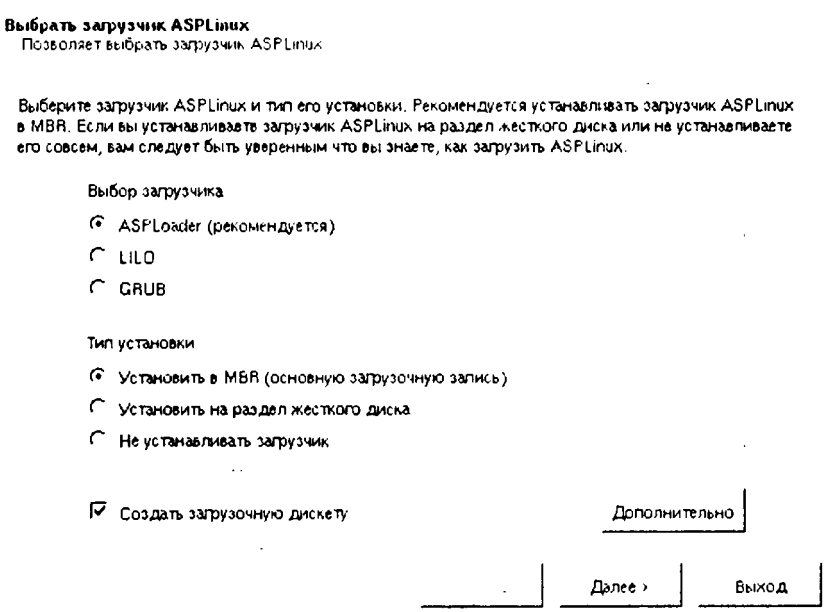

Рис. 2.6. Установка загрузчика Linux

В Linux есть множество загрузчиков, в моем дистрибутиве есть выбор между LILO (Linux Loader), GRUB (Grand Unified Bootloader) или ASP Loader (фир-

менный загрузчик ASP, который отличается от LILO только красотой). Я рекомендую выбрать LILO, хотя возможностей у него меньше, чем у GRUB, но их достаточно, и в настройке он проще. Устанавливайте его в MBR, если нет особой надобности загружать систему с дискеты.

Загрузчик при старте системы позволит выбрать, какую именно операционную систему вы хотите использовать: Windows, если он установлен на компьютере, или Linux (или любое его ядро, если их несколько).

Внизу окна (рис. 2.6) можно установить флажок **Создать загрузочную дискету**. Я рекомендую обязательно сделать это, если вы выбрали тип установки не в MBR. Загрузочная дискета может спасти, если загрузчик испорчен на диске или вообще отсутствует.

Двигаемся дальше. При наличии сетевой карты в компьютере вы увидите окно для ее выбора. Если нужный драйвер в списке не найдется, то можно оставить значение **none**. Это не значит, что сетевая карта не заработает, просто будет использоваться универсальный драйвер. В дальнейшем я рекомендую установить корректные драйверы, чтобы заставить сетевую карту работать на все 100 %.

На следующем шаге задаются сетевые настройки. Если в вашей сети используется единственный DHCP-сервер, то можно оставить все по умолчанию. Если в вашей сети несколько серверов или есть необходимость расставить адреса вручную, то уберите галочку в пункте **Настроить с помощью DHCP**.

Если вы не знаете, как работает протокол TCP/IP, то в качестве адреса для примера можете указать 192.168.77.1. Перейдите в поле **Шлюз**, и все остальные параметры будут заполнены автоматически. В поле **Маска** должно быть значение 255.255.255.0. В *разд. 3.6* мы поговорим о настройке сети, и вы узнаете, как можно изменить параметры подключения. В поле **Имя хоста** укажите имя, которое вы хотите задать своему компьютеру.

Последнее, что необходимо сделать, — это настроить вашу видеокарту для правильной работы в графическом режиме Windows. На рис. 2.7 показано окно мастера, позволяющего задать параметры видео. Ваша задача правильно выбрать видеокарту и характеристики экрана. Если вы ошибетесь, то сразу же познакомитесь с командной строкой Linux, но все же сможете решить проблему (в *разд. 2.7* мы рассмотрим этот вариант).

Не забудьте нажать кнопку **Тестировать**, чтобы проверить установленные настройки. Если все прошло удачно, можно поставить галочку в поле **Использовать графический ввод**, что позволит вам очутиться в приятном графическом мире Linux.

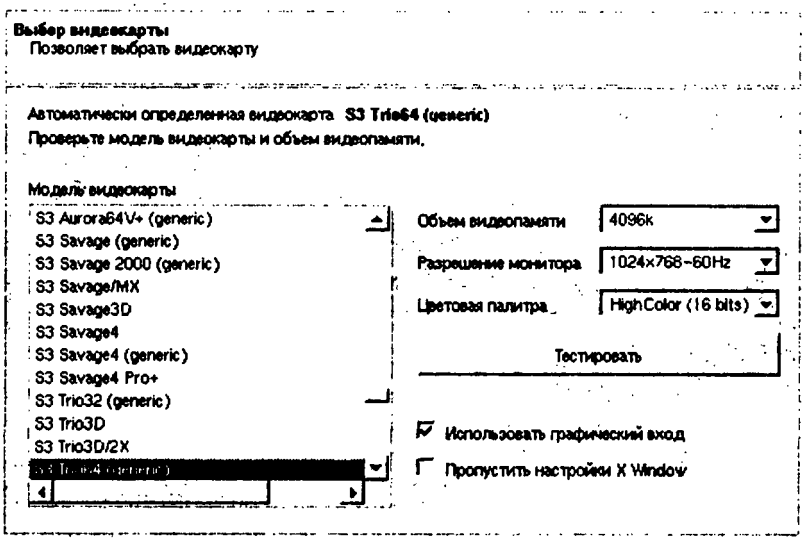

Рис. 2.7. Окно настройки видео

2.6. Пароль

Последнее, что нам нужно указать, — это пароль администратора системы (root). В Linux, как и в Windows XP Professional (не путать с Home Edition), нельзя входить без пароля, подобно Windows 9*x*. Вы обязательно должны указать имя пользователя и пароль, под которым будете работать, и в зависимости от ваших прав вам будет предоставляться доступ к определенным разделам и функциям ОС (рис. 2.8).

Программа установки проверяет только длину пароля, и для администратора эта величина должна быть не менее 6 символов. Так как пользователь root имеет полные права на систему, то пароль должен быть как можно более сложным, чтобы его нельзя было быстро подобрать.

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, читаемые слова или даты рождения. Такие пароли легко взламываются простым перебором по словарю, и это не отнимет много времени, если словарь хорошо составлен.

При создании пароля желательно генерировать случайные шифры, в которых будут символы разного регистра (прописные и строчные буквы), а также

цифры и различные допустимые символы (например, тире или подчеркивание). Длина пароля должна быть не менее 8 символов, а лучше более 12. Тогда для подбора хакеру потребуется намного больше времени.

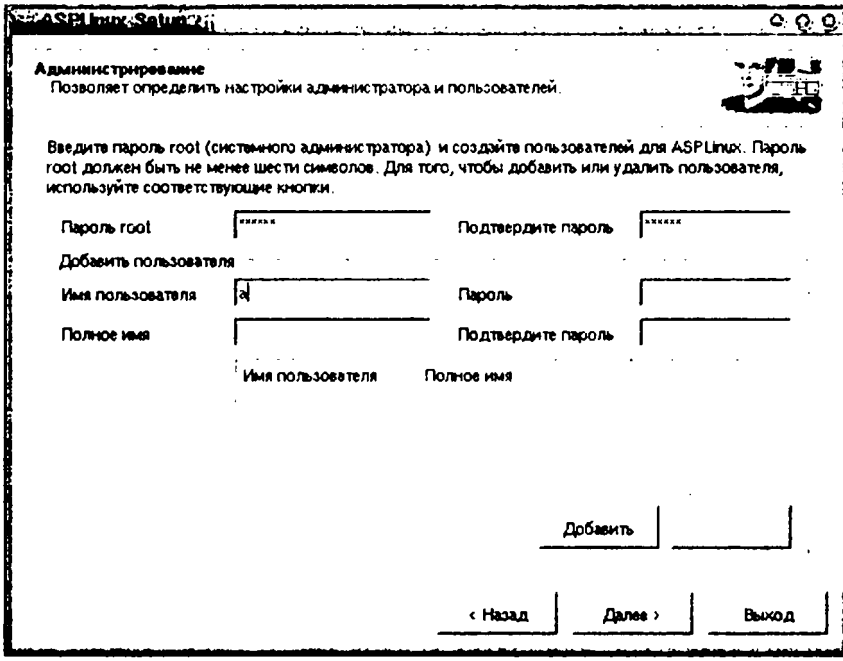

Рис. 2.8. Окно задания пароля

Когда нужно сгенерировать пароль, я запускаю какой-нибудь текстовый редактор и случайным образом набираю на клавиатуре любые символы в разном регистре. Как теперь использовать такую комбинацию? Лучше потратить пару дней на усвоение сложного пароля, чем лишиться важных данных.

Если не хочется запоминать что-то сверхсложное, то можно применить метод попроще, но и надежность полученного шифра будет ниже. Рассмотрим очень интересный способ генерации случайных паролей. Допустим, что вы хотите использовать слово generation. А что, оно достаточно длинное, но простое и может быть легко взломано по словарю. Как усложнить пароль? Посмотрите на клавиатуру и набирайте вместо букв слова generation символы, находящиеся немного выше. Например, прямо над "g" находится "t", а над "e" — "3" и т. д. Таким образом получится пароль t3h34q589h. Такой пароль запоминается легко, и по словарю подобрать его сложнее.

Вместо верхних можно взять буквы, находящиеся справа, и тогда пароль generation превратится в hrmrtsypm. Тоже нелегкая задача для хакера.

А если еще набрать некоторые из этих букв в верхнем регистре, то пароль усложнится сразу в два раза. Например, вы можете установить в верхнем регистре третью и восьмую буквы и получить hrMrtsyPm.

Вот таким простым способом можно соорудить легко запоминаемый, но сложный для подбора пароль.

Помимо этого вам предлагается завести новую учетную запись (добавить пользователя), под которой вы будете в дальнейшем работать с системой. Конечно же, можно использовать и root, но это не рекомендуется. Даже администраторы должны входить в систему как простые пользователи (возможно, с небольшим завышением прав на доступ к необходимым объектам) и только при крайней надобности переключаться на учетную запись `root`.

На этом установка завершена и можно перезагружать компьютер, вытащив CD-диски из привода.

2.7. Первый старт

При старте компьютера пока все будет происходить по-старому: тестирование памяти, определение дисков, отображение информации о системе. И вот тут появится окно загрузчика Linux, который мы определили в *разд. 2.5* (рис. 2.9).

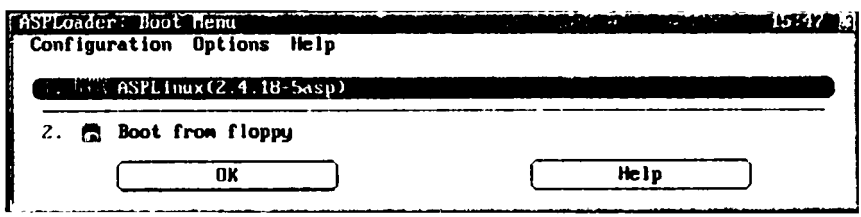

Рис. 2.9. Окно выбора варианта загрузки

В моем случае ОС устанавливалась на чистый компьютер, и возможны только два варианта: запустить Linux и загрузиться с дискеты. Впоследствии у вас может быть несколько ядер Linux, тогда список для выбора будет шире. Если у вас была установлена ОС Windows, то в этом окне можно будет увидеть еще и строку загрузки Windows. Таким образом, на одном компьютере могут вполне мирно существовать Windows и Linux. Хотя надо заметить, что Windows XP и старые загрузчики Linux не дружат. Видимо, XP не хотел терпеть конкурентов на одном компьютере и убивал LILO. Сейчас загрузчики Linux стали умнее и не дают себя в обиду.

В некоторых дистрибутивах, в том числе и Red Hat, еще можно встретить командный вид приглашения для загрузки системы. После старта компьютера появляется строка для ввода необходимой ОС:

```
LILO boot:
```

Если при установке вы выбрали загрузку Linux по умолчанию, то достаточно нажать клавишу <Enter>, иначе необходимо набрать на клавиатуре заголовок загружаемой ОС (в данном случае это будет слово "linux") и после этого уже нажать <Enter>. Вместо набора имени загружаемой системы можно его выбрать, воспользовавшись стрелками на клавиатуре или клавишей <Tab>.

Найдите загрузку Linux и облокотитесь в кресле в ожидании окна приглашения ввода пароля. На экране будет бежать белый текст на черном фоне с информацией о найденных устройствах, о версиях различных модулей и т. д.

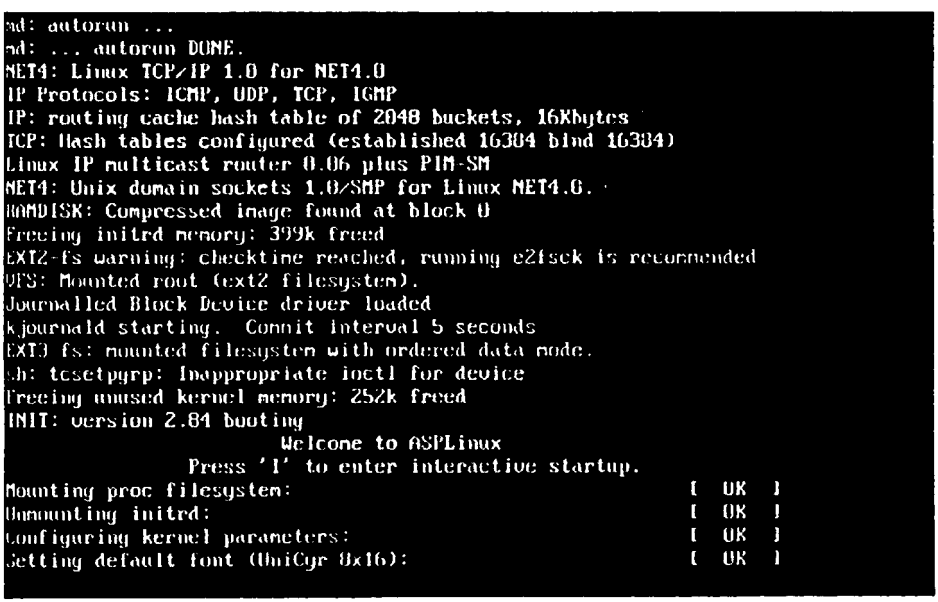

Рис. 2.10. Процесс загрузки Linux

После того как вы по центру экрана увидите надпись "Welcome to ASPLinux. Press 'I' to enter interactive startup" или "Добро пожаловать в ASPLinux. Нажмите 'I', чтобы перейти в интерактивный режим" (рис. 2.10), можно нажать кнопку <I>, тогда перед загрузкой очередного сервиса система будет просить подтверждение. Это очень удобная возможность, когда система нарушилась и какой-то сервис приводит к зависанию. Например, у меня очень часто после установки такое происходит с демоном sendmail (в *гл. 8* мы разберем проблему). Если это случилось, то ОС уже не может загрузиться. Для выхода из сложившейся ситуации просто перезапустите компьютер, войдите в интерактивный режим и на вопрос о загрузке sendmail ответьте "No".

ОС Linux — многопользовательская. Это значит, что с ней могут работать поочередно или одновременно несколько человек. Система должна знать, с кем она сейчас работает, поэтому после загрузки ОС вам предложат представиться с помощью указания имени пользователя и пароля. Первый параметр позволяет идентифицировать вас, а второй — обезопасить от нежелательного использования вашего имени другим пользователем.

Итак, идентификация пользователя в текстовом режиме выглядит, как строка-приглашение ввести имя:

```
localhost login:
```

Затем вы должны указать пароль, чтобы ОС смогла четко определить, что вы являетесь тем, за кого себя выдаете.

Если вы предпочли графический вход в систему, то появится окно, как на рис. 2.11. Обращаю ваше внимание, что окна могут отличаться в различных дистрибутивах Linux.

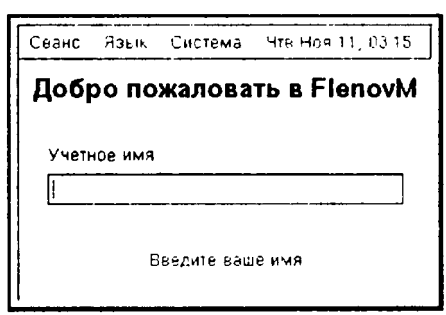

Рис. 2.11. Окно ввода имени и пароля

Прежде чем вводить имя, осмотритесь. В окне есть меню, состоящее из трех пунктов:

- **Сеанс** (Session) — позволяет выбрать графическую оболочку, которую вы хотите загрузить. Мы в основном будем работать с KDE или GNOME, как

наиболее распространенными. Что выберете вы, зависит от личных предпочтений;

- **Язык** (Language) — по умолчанию используется английский язык, но вы можете изменить его, например, на русский. ОС Linux поддерживает достаточно обширный список национальных языков, и он постепенно расширяется;
- **System** (Система) — в этом меню находятся команды перезагрузки и выключения компьютера.

В меню **Session** (Сеанс) есть еще один очень интересный пункт — **Failsafe**. Если ваша система настроена с ошибкой и графическая оболочка не может запуститься, то выберите этот пункт, и тогда до перехода в графический режим вы увидите окно терминала и в нем командную строку, с помощью которой можно исправить проблему.

Определив нужные параметры, введите имя пользователя и пароль, которые вы указали во время установки ОС, и добро пожаловать в графический мир Linux. Но под какой учетной записью работать? Во время установки мы задали пароль системного администратора (root) и добавили пользователя. Я настоятельно рекомендую выбрать пользовательскую запись, потому что root обладает всеми правами и может творить все, что угодно. Эта власть нередко приводила к плачевным последствиям, когда администраторы во время тестирования каких-либо параметров по ошибке уничтожали важные данные.

Работа под учетной записью `root` может облегчить взлом компьютера через простые на первый взгляд приложения. Допустим, что вы путешествуете по Web-сайтам. Запущенный браузер автоматически имеет те же права. И если в нем найдется уязвимость, позволяющая получить доступ к жесткому диску, то злоумышленник сможет воспользоваться этой лазейкой и, например, стереть всю информацию.

Если работать под пользовательской учетной записью, то уничтожить можно будет только те файлы, которые доступны этой записи. Системные файлы в таком случае в большей безопасности. При необходимости повышения прав вы всегда можете это сделать, зная пароль администратора. Для этого используется 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 указано тире. 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.

ОС Linux — многопользовательская система и поддерживает несколько терминалов. По умолчанию вы входите в первый терминал. Для того чтобы переключиться на другой, нужно нажать клавишу <Alt> и одну из клавиш <F1>—<F6>. В ответ на это перед вами появится чистый экран с приглашением ввести пароль, который может быть различным в каждом терминале.

Использование терминалов — очень удобная возможность. В одном вы запускаете программу, которая требует много времени для выполнения, и между тем переключаетесь на другой терминал и продолжаете работу с компьютером. В любой момент можно вернуться на первый терминал и посмотреть на ход исполнения программы.

Если вы перед запуском установили графический режим, то сразу после загрузки перед вами откроется окно, в котором система предложит использовать выбранную оболочку по умолчанию. Это значит, что при следующем старте, если не указано иного, будет запущена именно эта оболочка.

Если вы выбрали текстовый режим, то в любой момент можете переключиться на графический, набрав команду `startx`. В этом случае графическая оболочка будет загружена автоматически, без приглашения на ввод пароля, потому что вы уже идентифицировали себя в текстовой консоли.

Текстовое приглашение на вход в систему может появиться и в случае ошибки конфигурации. В этом случае выполнение команды `startx` ни к чему не приведет, т. к. графическая оболочка не сможет загрузиться. Тогда придется произвести настройку графики заново. Большинство информации о конфигурации Linux находится в текстовых файлах и нередко приходится править их вручную. В данном случае также используются текстовые файлы, но ручное редактирование необязательно, потому что есть специальная и удобная утилита setup. Наберите в командной строке команду `setup`, и перед вами откроется окно, как на рис. 2.12. Выберите пункт **X Configuration**. Система, скорее всего, сама определит наличие видеокарты и необходимые драйверы, а вот с монитором могут быть проблемы. Чаще всего приходится самостоятельно указывать его тип и поддерживаемые видеорежимы.

Кстати, о видеорежимах. В списке будут представлены все возможные значения, и вы можете указать любое количество. Я рекомендую остановиться только на том, который для вас является максимально удобным. После настройки обязательно протестируйте выбранный графический режим на предмет работоспособности. Если все пройдет удачно, то программа предложит использовать графический режим для входа в систему.

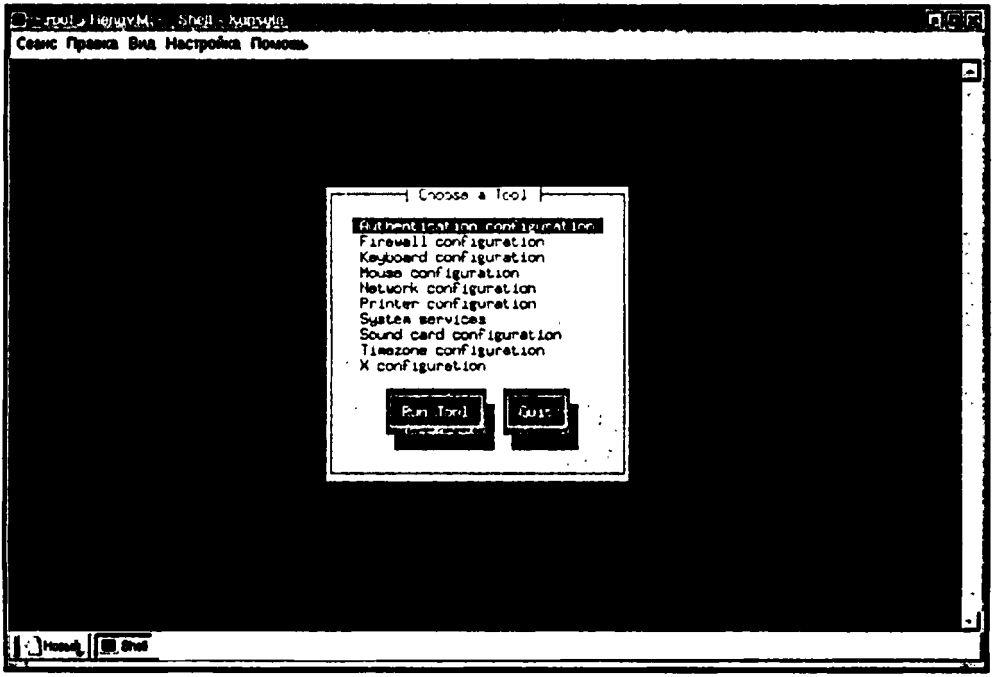

Рис. 2.12. Главное окно утилиты Setup

Во время настройки можно использовать мышь, но если она отсутствует, то достаточно и клавиатуры. Для перемещения между кнопками используйте клавишу <Tab>, для выделения пунктов меню — пробел, а передвижение внутри списка осуществляется стрелками <↑> и <↓>.

С настройкой видеокарты (и то на чипсете S3, собранной в Китае) у меня последний раз возникали проблемы года три назад еще на Red Hat 6.1. Современные дистрибутивы без проблем определяют железо, особенно если оно собрано именитым брендом.

Надеюсь, что вам удалось войти в графический режим. Если во время загрузки произошла ошибка, завис компьютер или изображение на экране стало искаженным, графическую оболочку можно остановить нажатием клавиш <Ctrl>+<Alt>+<Backspace>. Таким образом, вы переключитесь в текстовый режим.

Если загрузка графической оболочки прошла удачно, то самое время познакомиться с окном терминала, в котором можно выполнять команды как в текстовом режиме. Домашнему пользователю это не обязательно, потому что ему нужны игры и офисные программы. А для администрирования и тонкой настройки Linux терминал просто необходим.

Рис. 2.13. Рабочий стол в графической оболочке KDE

На рис. 2.13 показано окно графической оболочки KDE, а на рис. 2.14 — оболочка GNOME.

В обеих оболочках в нижней части Рабочего стола находится Панель задач. Здесь располагаются:

- кнопка вызова главного меню (самая левая кнопка) — аналог кнопки "Пуск" в Windows. В этом меню можно найти все установленные на компьютере программы и утилиты;
- кнопка быстрого вызова основных команд — на обоих рисунках выделена красным овалом, при нажатии осуществляется вызов окна терминала;
- пустое поле — здесь будут располагаться кнопки запущенных программ. Кнопки задач можно использовать для переключения между работающими приложениями.

Откройте окно терминала, чтобы познакомиться с его внешним видом. В большинстве дистрибутивов по умолчанию это окно с черным фоном и текстом белого цвета. Сразу после запуска перед вами появится приглашение ввода команд, которое может выглядеть примерно следующим образом:

```
[root@Flenov root]:
```

Что это значит? Сначала идет имя пользователя, под которым вы вошли в систему (в данном случае это root). После знака @ следуют имя компьютера и пробел, за которым стоит имя текущей папки.

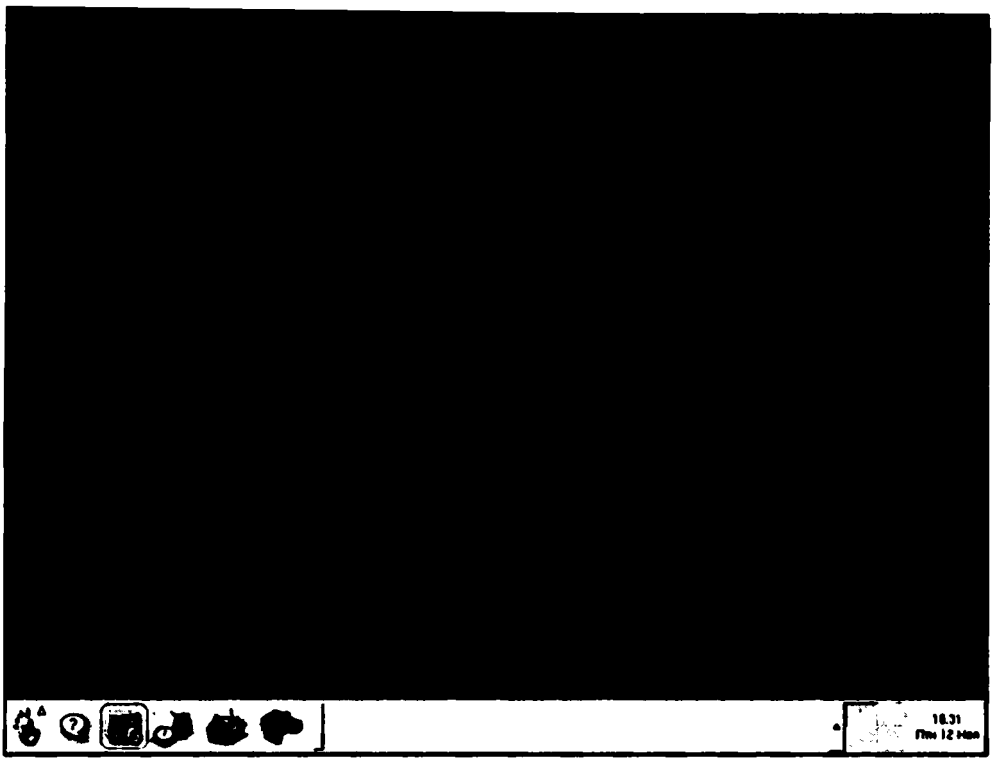

Рис. 2.14. Рабочий стол в графической оболочке GNOME

2.8. Мы в системе

Мы вошли в систему, и ОС запускает для нас уникальное окружение, которое определяется по имени пользователя. Среда включает пользовательские директории, настройки и командную оболочку.

Пользовательские директории находятся в каталоге **/home** и имеют название, совпадающее с именем пользователя. Например, для пользователя root есть подкаталог **/home/root**. В этой папке вы можете хранить свои файлы.

Когда вы входите в систему, то текущей директорией становится ваша. Так для пользователя root таковой станет **/home/root**, и все команды будут выполняться в этом каталоге, пока мы не изменим его.

В Linux существует несколько командных оболочек, и каждая из них обладает своими возможностями. Чаще всего пользователи применяют оболочку под названием bash (Bourne Again Shell). Мы будем рассматривать именно ее.

Первые шаги мы уже сделали, теперь приступим к более глубокому знакомству с Linux. По мере изучения вы сможете настроить систему более тонко и сделаете ее максимально быстрой, эффективной и безопасной.

Когда я начинал осваивать Linux и установил ее в первый раз, то потом долго не мог правильно выключить, потому что не знал, какая для этого используется команда. Я взял у знакомого книгу по этой ОС, но даже в ней описание процесса выхода было где-то в середине, и пока я нашел нужную команду, пришлось месяц обходиться без корректного завершения работы ОС и выключать питание компьютера.

Спешу поделиться с вами накопленным опытом. Для выхода из системы используется команда `shutdown`. После нее указывается, что именно нам нужно: `-r` для перезагрузки или `-h` для выключения компьютера. И в конце команды можно задать время, через которое должно начаться выполнение команды (в частности, `now` — сейчас).

Например, для немедленной перезагрузки системы наберите команду:

```
shutdown -r now
```

Для немедленного выключения компьютера следует ввести:

```
shutdown -h now
```

Обязательно используйте эти команды. Если выключить питание с помощью кнопок на системном блоке, то можно потерять данные, которые были загружены в оперативную память, но не сохранены на жестком диске.

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

Чтобы выйти из системы в графическом режиме, нажмите кнопку вызова главного меню и выберите пункт **Выйти из системы**. Это заставит графическую оболочку выгрузится, затем снова появится окно-приглашение ввести пароль, где можно войти в систему под новым именем или перезагрузить компьютер.

2.9. Подсказки

Некоторые операционные системы славятся своей простотой и хорошими подсказками. Но я не устану повторять, что производительность, надежность и простота не всегда совместимые вещи. ОС Linux содержит множество команд, и у каждой из них может быть 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 свой ключ, который чаще всего заставит вывести на экран короткую подсказку об использовании программы.

Для получения более подробной информации нужно воспользоваться следующей конструкцией:

```
man имя
```

Имя — это название команды или программы, а `man` выводит на экран информацию об объекте. Например, чтобы увидеть описание команды `shutdown`, выполните `man shutdown`.

Для выхода из программы помощи нужно набрать на клавиатуре <-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, и программа `man` завершит работу.

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

Если вы устанавливаете лицензионный вариант ОС, то в коробке можно найти документацию по установке и руководство пользователя. Чаще всего эта информация поверхностная и позволяет сделать только минимальные настройки. Но некоторые дистрибутивы содержат очень подробную документацию или даже целые книги.

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 ОС и сервиса. Если вы будете их придерживаться, то сможете построить действительно защищенную систему (сервер или даже сеть из компьютеров и серверов).

2.10.1. Запрещено то, что не разрешено

Когда вы настраиваете параметры доступа, то необходимо придерживаться принципа минимализма, который заключается в следующем:

1. Необходимо запускать минимум возможного. Это касается не только сервисов в целом, но и их составляющих. Например, вам нужен Web-сервер и для этого чаще всего устанавливается Apache. Эта программа включает

в себя очень много возможностей, среди которых поддержка интерпретируемых языков PHP и Perl, но, как правило, программисты сайтов используют только один из этих языков, поэтому нет смысла разрешать оба. Если сайт проектируется с помощью PHP, то следует удалить Perl, и наоборот. Ну а если ваш сайт использует сразу два языка сценариев, увольте своих программистов, потому что в разнородных системах намного сложнее построить безопасность.

2. Следует разрешать минимум необходимого. Большинство администраторов не любят заниматься какими-либо настройками, поэтому открывают полный доступ на все, что только может пригодиться. Но слово "может" не совместимо с безопасностью. Возможность, которую вы откроете пользователю, на самом деле может ему не понадобиться, а вот злоумышленник благодаря лишней лазейке может натворить много бед. Например, на клиентских компьютерах может быть открыта какая-то папка для всеобщего доступа. Это мотивируется тем, что пользователям может потребоваться обмен данными. А если нет?

При рассмотрении конфигурирования ОС и ее сервисов я буду часто напоминать об этом правиле, и при разборе примеров будем отталкиваться именно от полного запрета.

2.10.2. Настройки по умолчанию

Настройки по умолчанию предусмотрены только для обучения и чаще всего открывают абсолютно все возможности, чтобы вы могли оценить мощь программ. Это значит, что будет разрешено абсолютно все, а это уже нарушает второе рассмотренное правило.

Если используется не так много команд и параметров, то лучше заняться конфигурацией программы с чистого листа. Если процедура достаточна сложная (ярким примером является Sendmail), то лучше всего отталкиваться от настроек по умолчанию, добавляя, изменяя или удаляя ключи. Даже не пытайтесь в этом случае настраивать систему с нуля. В процессе конфигурирования обязательно что-то забывается и тем самым повышается вероятность ошибки. Сложность конфигурационных файлов в том, что они имеют текстовый формат и все имена параметров нужно писать четко и полностью. Если ошибиться хотя бы в одной букве, параметр будет воспринят неверно, и появятся сбои в работе ОС или сервиса.

Когда 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 директорий. Эта особенность имеет значение и для некоторых конфигурационных файлов.

2.10.3. Пароли по умолчанию

Многие сервисы во время установки прописывают пароли по умолчанию. В ОС Linux эта проблема стоит особо остро, потому что программы инсталляции используют RPM-пакеты и чаще всего даже не предлагают их сменить. Я бы на месте разработчиков вообще запретил запуск сервисов с пустым или неизмененным паролем.

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

Прежде чем сдавать систему в эксплуатацию, убедитесь, что изменены все пароли. Снова пример с MySQL. Администраторы редко используют его, а только устанавливают. 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 так делаю в надежде, что за паролями проследит администратор, а тот надеется на меня, и получается, что мы оба забываем.

После того, как несколько раз система оказалась уязвимой, я разрабатываю программу под своей учетной записью и с измененным паролем. Лучше это сделать дважды, чем забыть выполнить совсем.

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. Производители, не долго думая, чаще всего устанавливают имя Admin, а пароль оставляют пустым. Это большое упущение. Я бы на их месте для пароля по умолчанию использовал серийный номер устройства. В этом случае хакер не сможет его подобрать. Хотя и серийный номер не является абсолютной защитой, потому что если хакер увидит устройство своими глазами, то он легко сможет вычислить и пароль.

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

2.10.4. Универсальные пароли

Производители BIOS раньше устанавливали в свои чипы универсальные коды доступа, которые позволяли войти в систему, не зная основной пароль, который установил администратор. Например, в одной из версий BIOS ком-

пании AWARD использовался универсальный шифр AWARD_SW. Начиная с версии 4.51, такая "услуга" отсутствует.

Если у вас есть возможность отключить использование универсального пароля, *то сделайте* это незамедлительно. В противном случае смените оборудование или программу, иначе нет смысла пытаться сделать вашу систему защищенной от вторжения.

2.10.5. 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, сетевые экраны, а они расходуют ресурсы процессора. И чем больше дополнительных служб включено, тем больше лишних затрат.

Каждый ресурс может быть настроен по-разному. Например, в режиме журналирования можно записывать в журналы только основную информацию, что позволит уменьшить нагрузку на жесткий диск, но увеличит вероятность того, что какая-то атака пройдет незамеченной. А можно сделать так, что в журнал будут попадать абсолютно все сообщения. В этом случае повышается расход ресурсов, и у хакера появляется шанс удачно произвести атаку DoS.

Во время конфигурирования сервера и его сервисов вы должны исходить из принципа необходимой достаточности. Следует принимать все меры, чтобы сервер или компьютер чувствовал себя в безопасности, но при этом работал как можно производительнее. Чтобы убедиться в нормальном балансе этих параметров, после окончания конфигурирования необходимо заставить сервер работать при максимально возможной загрузке (определяется планируемым количеством обрабатываемых запросов в минуту умноженное на 2). Если сервер справится с поставленной задачей и сможет обработать все запросы клиентов, и при этом еще останется запас в производительности процессора, то можно вводить машину в эксплуатацию. Иначе необходимо изменять конфигурацию или наращивать мощность компьютера.

ГЛАВА 3

Добро пожаловать в Linux

В этой главе мы начнем знакомиться с самим Linux. Надеюсь, что вы уже установили систему, потому что все, что мы будем рассматривать, правильнее всего тут же проверять на практике. Только так материал будет лучше откладываться в памяти и усваиваться.

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. Многие авторы почему-то ограничиваются рассмотрением только текстового режима в консоли. Это пугает тех пользователей, которые привыкли к Windows и интуитивно понятному интерфейсу. Мы будем разбирать одновременно оба режима. И все же, консоли будет уделяться достаточно много внимания, потому что зачастую с ее помощью можно быстрее, нежели через графические утилиты, решить какие-либо проблемы. Я постараюсь показать вам преимущество консоли перед курсором мыши. Дело в том, что серверы на предприятиях должны стоять в отдельной комнате и, возможно, даже без монитора. Управление происходит через удаленную консоль, и визуальные возможности Linux не используются. Тогда зачем загружать тяжелые графические библиотеки, файлы и другие ресурсы? Это же пустое расходование памяти!!! Не лучше ли ее освободить для более полезных вещей.

Графический режим необходим для работы с пользовательскими утилитами. Он также может быть полезен на первоначальном этапе настройки сервера. А если учесть, что не все компьютеры на базе Linux являются серверами, и домашние станции тоже могут работать на этой ОС, то удобный графический интерфейс необходим.

Как видите, возможность работать в двух режимах — это преимущество Linux, а не недостаток. Если бы в Windows можно было выгрузить из памяти графическую оболочку и оставить только командную строку, то вы смогли бы сэкономить драгоценную память и повысить надежность этой ОС. Когда

не работают графические библиотеки, то и проблемы с ними отсутствуют. Сколько раз мы видели синие экраны с ошибками в драйвере видеокарты? В консоли Linux этого произойти не может.

Если вы настраиваете домашний компьютер (или маленькую сеть), то графическую оболочку можно оставить. Но если это промышленный сервер, требующий максимальной доступности, то я рекомендую оставить компьютер в текстовом режиме, чтобы обезопаситься от лишних сбоев и повысить производительность.

3.1. Файловая система

Прежде чем перейти к настройкам системы, нам нужно познакомиться поближе с файловой системой Linux. О структуре мы уже немного поговорили в *разд. 2.3*, когда разбивали жесткий диск. В табл. 2.1 были перечислены разделы, которые можно создать в Linux, а это не что иное, как основные папки.

Теперь, наверное, нужно было бы перечислить команды, с помощью которых можно управлять директориями и файлами, а также просматривать и редактировать их. Но мы сделаем это чуть позже, а сейчас я хочу показать одну лишь программу Midnight Commander (MC). Это лучшее средство для решения всех описанных выше задач. Программа присутствует в большинстве дистрибутивов, в том числе и в Red Hat. Для ее запуска наберите в командной строке `mc` и нажмите клавишу <Enter>. Постепенно мы будем знакомиться с этой утилитой, и вы полюбите ее за удобство и мощь, а сейчас рассмотрим только основные возможности.

На рис. 3.1 изображено окно терминала, в котором запущена программа MC. Окно состоит из двух независимых панелей, в каждой из которых вы можете видеть файлы и папки текущей директории (имена папок начинаются с символа слэш). Для перемещения между панелями используется клавиша <Tab>.

С правой стороны показана корневая папка. Это самый верхний уровень вашей файловой системы. Посмотрите на список папок в этой панели. Большинство названий нам знакомо по табл. 2.1. Каждая из этих папок может находиться в собственном разделе жесткого диска, если при установке вы его создали. Но даже в этом случае в файловой системе вы будете видеть все как одно целое.

В *разд. 2.3.3* мы говорили про корневой каталог, который в Linux обозначается как знак "/". Именно он является вершиной пирамиды в иерархии всех каталогов. Например, папки пользователя находятся в каталоге **/home**. Тогда **/home/flenov** будет определять путь к подкаталогу пользователя flenov. Чтобы попасть в эту директорию, нужно навести на нее курсор и нажать <Enter>.

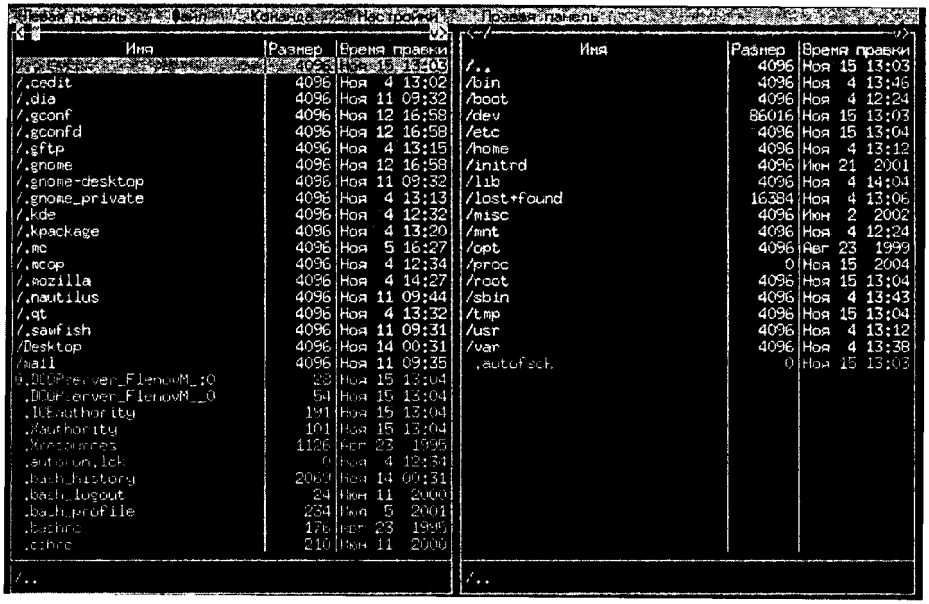

Рис. 3.1. Окно терминала с запущенной программой Midnight Commander

В списке папок и файлов самой первой всегда стоит папка с именем **/...** Реально такого каталога не существует. Это указатель на родительскую директорию, с помощью которой вы можете попасть на уровень выше. Например, вы находитесь в подкаталоге **/home/flenov**. Если войти в папку **/..**, то вы поднимитесь на предыдущий уровень и окажетесь в директории **/home**.

Внизу окна MC (см. рис. 3.1) можно увидеть строку-приглашение для ввода команд. Эта та же строка, что мы видели в терминале, и позволяет выполнять те же директивы. Еще ниже расположена строка меню с подсказками о назначении клавиш <F1>—<F10>:

- 1 (Помощь) — вызов файла помощи по программе;
- 2 (Меню) — вызов меню основных команд MC;
- 3 (Просмот) — просмотр выделенного файла;
- 4 (Правка) — редактирование выделенного файла во встроенном текстовом редакторе;
- 5 (Копия) — копирование выделенного файла или папки. Если выделить файл и нажать клавишу <F5>, то появится окно подтверждения копирова-

ния. По умолчанию операция выполняется в текущую директорию противоположной панели программы MC;

- 6 (Перемес) — переместить выделенные файлы или папки. По умолчанию файл будет перенесен в директорию, являющуюся текущей для противоположной панели программы MC;
- 7 (НвКтлог) — создать новый каталог в текущем;
- 8 (Удалить) — удалить выделенные файлы и папки;
- 9 (МенюMC) — вызвать меню программы MC, которое находится вверху окна;
- 10 (Выход) — выход из программы.

Файлы и папки, имена которых начинаются с точки, являются конфигурационными. Будьте осторожны при их перемещении и редактировании. Эти файлы нуждаются в максимальной защите, но об этом мы поговорим позже в разных разделах книги.

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.

pwd

Эта команда выводит на экран полный путь к текущему каталогу. С ее помощью вы можете в любой момент узнать, где находитесь.

ls

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

```
ls -a
```

Если мы хотим увидеть не только имена (сжатый формат), но и полную информацию о каталоге, нужно добавить ключ `-l`. В результате мы должны выполнить команду:

```
ls -al
```

Но такая команда отобразит файлы текущей директории, и не факт, что мы сейчас находимся, например, в каталоге **/etc**, который надо просмотреть. Что-

бы увидеть именно его, после ключей (можно и до них) нужно указать требуемую папку:

```
ls -al /etc
```

Примечание

Более подробную информацию о команде `ls` можно получить из справочной системы. Для этого выполните команду `man ls`.

Рассмотрим результат вывода команды `ls -al`:

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

По умолчанию список файлов выводится в несколько колонок. Разберем их на примере первой строки:

- `drwx------` — права доступа. Их мы подробно рассмотрим в *гл. 4*. Сейчас вам только нужно знать, что если первая буква "`d`", то это директория;
- цифра `3` — указывает количество жестких ссылок;
- `Flenov` — имя пользователя, являющегося владельцем файла;
- `FlenovG` — группа, которой принадлежит файл;
- `4096` — размер файла. Для директории это значение отсутствует, т. к. не устанавливается ее размер;
- дата и время последних изменений файла;
- имя файла.

cat

Команда позволяет вывести на экран содержимое указанного в качестве аргумента файла. Например, вы хотите просмотреть текстовый файл need.txt. Для этого нужно выполнить команду:

```
cat need.txt
```

Но это справедливо, если файл находится в текущей директории. А если нет? В этом случае можно указать полный путь:

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

tac

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

cd

Эта команда позволяет сменить текущий каталог. Для этого необходимо в качестве параметра задать нужную папку:

```
cd /home/flenov
```

Если вы находитесь в каталоге **/home** и хотите внутри него перейти в подкаталог **flenov**, то достаточно набрать только имя папки **flenov**:

```
cd flenov
```

Если нужно переместиться на уровень выше, например, из подкаталога **/home/flenov** в каталог **/home**, нужно выполнить команду:

```
cd ..
```

Как мы знаем, папка с именем из двух точек указывает на родительский каталог. Если перейти на нее, то мы попадем на предыдущий уровень.

cp

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

1. Копирования содержимого 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).

2. Копирования файла в другой каталог:

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

   или

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

   Обратите внимание, что в этом случае в папке назначения файл может быть как с новым, так и со старым именем.

3. Копирование несколько файлов в новый каталог. Для этого нужно перечислить все файлы в источнике и последним параметром указать папку назначения:

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

В этом примере файлы **/home/root/need.txt** и **/home/root/need22.txt** будут скопированы в директорию **/home/new**. Можно копировать файлы и из разных каталогов в один:

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

В этом примере файлы **/home/root/need.txt** и **/home/flenov/need22.txt** будут скопированы в директорию **/home/new**.

4. Копирование группы (всех) файлов каталога.

   А что если надо скопировать все файлы, начинающиеся на букву "n" из одной директории в другую? Неужели придется их все перечислять? Нет, достаточно указать маску `n*`, где звездочка заменяет любые символы, начиная со второго:

   ```
   cp /home/root/n* /home/new/
   ```

   Если нужно скопировать все файлы, имена которых начинаются символами "ra" и заканчиваются буквой "t", то маска будет выглядеть как `ra*t`.

mkdir

Создание новой директории. Например, если вы хотите создать подкаталог **newdir** в текущей директории, то нужно выполнить команду:

```
mkdir newdir
```

rm

Команда позволяет удалить файл или директорию (должна быть пустая):

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

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

- `-d` — удалить директорию;
- `-r` — рекурсивно удалять содержимое директорий;
- `-f` — не запрашивать подтверждение удаляемых файлов. Будьте внимательны при использовании этого параметра, потому что файлы будут удаляться без каких-либо дополнительных вопросов. Вы должны быть уверены, что команда написана правильно.

Пример удаления директории:

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

df

Эта команда позволяет определить свободное место на жестком диске или разделе. Если устройство не указано, то на экран выводится информация о смонтированных файловых системах.

Пример результата выполнения команды:

```
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` — монтировка файловой системы.

mount

Команда предназначена для монтирования файловых систем. Она достаточно сложна, и ее используют системные администраторы.

Если вы работали с ОС Windows, то скорей всего привыкли к тому, что дискеты, CD-диски и другие съемные носители становятся доступными сразу же, как только вы поместили их в устройство чтения. В Linux это не так, и многие не могут сжиться с этой особенностью. К таким пользователям отношусь и я, т. к. до сих пор не могу привыкнуть, что нужно выполнять дополнительные команды, хотя и прекрасно понимаю, что они необходимы.

Итак, чтобы CD-ROM стал доступным, надо выполнить команду `mount`, указав в качестве параметра устройство **/dev/cdrom**:

```
mount /dev/cdrom
```

После этого содержимое CD можно посмотреть в директории **/mnt/cdrom**. Получается, что файлы и директории диска как бы сливаются с файловой системой.

Почему именно в директорию **/mnt/cdrom** подсоединяется CD-ROM? Секрет заключается в том, что для подключения CD-ROM нужно намного больше данных, чем дает одна команда `mount dev/cdrom`. Эти сведения хранятся в двух файлах, уже имеющихся в ОС и описывающих основные устройства и

параметры по умолчанию — файлы fstab и mtab. Давайте по очереди разберем эти файлы.

Для начала взглянем на fstab:

```
# /etc/fstab: static file system information.
#
# <file system> <mount point> <type> <options> <dump> <pass>
/dev/hda2  /       ext3 defaults,errors=remount-ro        0   1
/dev/hda1  none    swap sw                                0   0
proc       /proc   proc defaults                          0   0
none       /dev/shm  tmpfs defaults                       0   0
none       /dev/pts/ devpts gid=5,mode=620                0   0
/dev/cdrom /mnt/cdrom iso9660 noauto,owner,kudzu,ro       0   0
/dev/fd0   /mnt/floppy auto noauto,owner,kudzu            0   0
```

Файл содержит строки для основных дисков. Каждая запись состоит из 6 колонок. Обратите внимание на первую строку. Здесь описывается подключение диска **hda2**. В моей файловой системе это основной диск, поэтому второй параметр — "/". Это значит, что диск будет монтирован как корневой. Третья колонка описывает файловую систему, в данном случае это Ext2. Параметр `rw` указывает на то, что устройство доступно для чтения и записи.

Предпоследняя строка в файле описывает устройство CD-ROM. Посмотрите внимательно на второй параметр **/mnt/cdrom**. Вот откуда берется путь к содержимому CD-диска. Четвертая колонка содержит опции монтирования, в которых можно описать параметры безопасности. В данном случае для CD-ROM здесь указано несколько опций: `noauto`, `owner`, `kudzu`, `ro`. Очень важным здесь является параметр `ro`, который говорит о возможности только чтения CD-ROM. Вполне логично установить этот параметр для всех приводов и устройств, с помощью которых хакер сможет снять информацию с сервера.

Файл mtab имеет примерно такое же содержимое:

```
# <file system><mount point> <type> <options>    <dump> <pass>
/dev/hda2      /             ext3   rw,errors=remount-ro 0  0
proc           /proc         proc   rw                   0  0
none           /dev/shm      tmpfs  rw                   0  0
none           /dev/pts      devpts rw,gid=5,mode=620    0  0
none /proc/sys/fs/binfmt_misc binfmt_misc rw 0 0
/dev/cdrom /mnt/cdrom iso9660 ro,nosuid,nodev 0 0
```

Если вы создали какие-то разделы на отдельных дисках, то сможете настраивать и их. Я вам рекомендовал выделить таким образом раздел **/home** с пользовательскими директориями. Если вы так и сделали, то в файле может быть еще одна строка примерно следующего вида:

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

Посмотрим на четвертый параметр. В нем содержатся опции монтирования, которыми можно управлять для повышения безопасности системы. Они перечислены через запятую. В нашем примере это `rw,errors=remount-ro`. В качестве опций монтирования дополнительно можно использовать:

- `noexec` — запрет выполнения файлов. Если вы уверены, что в разделе не должно быть исполняемых файлов, то можно использовать эту опцию. Например, в некоторых системах директория **/home** должна хранить только документы. Чтобы хакер не смог записать в этот раздел свои программы, с помощью которых будет происходить взлом, добавьте этот параметр. Точнее сказать, программы поместить можно будет, 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 поговорим в *разд. 4.5*;
- `nodev` — запрещает использование файлов устройств;
- `nosymfollow` — запрещает использование мягких ссылок.

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

Использование 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. А таковым всегда является раздел с директорией **/bin** и другие каталоги, которые содержат необходимые для работы программы.

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`. Если для запуска сценария использовать командную строку, то вы получите сообщение о нарушении прав доступа. Но программа выполнится, если написать следующую команду:

```
perl file.pl
```

Несмотря на то, что `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 (дозволенная операция) и выполняет его в своем адресном пространстве.

Вспомните описание файла mtab, где для CD-ROM стоит запрет на использование SUID- и SGID-программ. То же самое необходимо сделать, как минимум, с разделами **/home** и **/tmp**. Тогда пользователи не смогут создавать

в своих директориях привилегированные программы, что позволит предотвратить большое количество возможных атак.

Итак, давайте попробуем смонтировать CD-ROM в другую директорию. Для этого сначала создадим ее:

```
mkdir /mnt/cd
```

Теперь выполним команду

```
mount /dev/cdrom /mnt/cd
```

Если на вашем компьютере установлено две ОС — Windows и Linux, то диск, скорее всего, содержит файловую систему FAT32 или NTFS. Следующие команды позволяют подключить FAT32 к Linux:

```
mkdir /mnt/vfat
mount -t vfat /dev/hda3 /mnt/vfat
```

Первая команда создает директорию **/mnt/vfat**, куда будет подключаться диск с FAT32.

Во второй команде происходит монтирование диска **/dev/hda3**. Будем считать, что как раз он и принадлежит Windows. Ключ `-t` позволяет указать тип подключаемой файловой системы. Это обязательно. Для CD-ROM мы этого не делали только потому, что вся необходимая информация есть в файле **/etc/fstab**. Файловая система указана в параметре `vfat`. Это имя для FAT32, которое используется в Linux.

Более подробно о работе команды можно узнать на страницах документации (`man mount`).

umount

Когда вы подключили к файловой системе CD-ROM, то это устройство блокируется, и диск нельзя вытащить, пока он не будет размонтирован. Для этого используется команда `umount`.

Например, следующая команда позволяет размонтировать CD-ROM:

```
umount /dev/cdrom
```

fdformat

Перед использованием дискет их нужно отформатировать. В ОС Linux для этого используется команда `fdformat`.

tar

По ходу изложения данной книги мы иногда будем устанавливать различные программы, часть из них поставляется в виде архивов tar.gz. Чаще всего это

программы, хранимые в исходных кодах. Для разархивирования такого файла нужно выполнить команду:

```
tar xzvf имяфайла.tar.gz
```

Как правило, после выполнения команды в текущей директории 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 *гл. 13*, когда будем рассматривать резервирование данных. Сейчас же нам достаточно уметь распаковывать пакеты, чтобы устанавливать дополнительные программы и утилиты сторонних разработчиков.

rpm

В настоящее время большинство программ поставляются уже не в исходных кодах, а в виде RPM-пакетов. Их установка намного проще, т. к. программы в них уже скомпилированы. Если вы используете MC, то выберите RPM-пакет и нажмите клавишу <Enter>. Таким образом вы войдете в него как в директорию и увидите содержимое.

Каждый пакет обязательно содержит исполняемый файл install. Запустите его для установки программы.

Если вы не используете MC, то 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 используется 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, после выполнения команды на экране выведется:

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

3.1.2. Безопасность файлов

В *гл. 4* мы будем подробно говорить о правах доступа. Это основа обеспечения безопасности, но и только, и надеяться на это нельзя. Необходимы дополнительные инструменты сохранения целостности системы, или, по крайней мере, вы должны следить за изменениями основных объектов ОС — файлами. В них хранится информация, а именно она необходима взломщикам. Хакеры стремятся прочитать, изменить или даже уничтожить информацию, поэтому вы должны уметь ее контролировать.

Дата изменения

Самый простейший способ контроля — наблюдение за датой редактирования. Допустим, что взломщик проник в вашу систему в 10:30. Чтобы узнать, что было изменено злоумышленником, можно запустить поиск всех файлов, у которых дата корректировки больше этого времени. Вроде легко, но не очень эффективно, потому что дату можно изменить с помощью команды `touch`. В общем виде команда выглядит следующим образом:

```
touch параметры ММДДччммГГ файл
```

Прописными буквами показаны параметры даты, а строчными — время. Формат немного непривычный, но запомнить можно. Год указывать необязательно, в этом случае будет использоваться текущий.

Рассмотрим пример. Допустим, что вы хотите установить на файл **/etc/passwd** дату изменения `21 января 11:40`. Для этого выполняем следующую команду:

```
touch 01211140 /etc/passwd
```

Теперь воспользуйтесь командой `ls -l /etc/passwd`, чтобы убедиться, что дата и время изменения установлены верно.

С помощью команды `touch` можно и создавать файлы, сразу же указывая необходимую дату.

Несмотря на то, что дата корректировки легко изменяется, хакер может забыть или просто не успеть сделать это, а, возможно, ему просто не хватит прав.

Итак, найти все файлы, дата изменения которых больше 21 января 11:40 2005 года, можно следующим образом:

```
touch 0121114005 /tmp/tempfile
find /etc \(-newer /tmp/tempfile \) -ls
find /etc \(-cnewer /tmp/tempfile \) -ls
find /etc \(-anewer /tmp/tempfile \) -ls
```

В первой строке мы создаем файл во временной директории **/tmp** с необходимой датой изменения, по которой и будет происходить сравнение.

Следующие три строки производят поиск файлов. Каждая из них имеет следующую структуру:

```
find директория \( -сравнение файл \) -ls
```

Рассмотрим по частям эту строку:

- `find` — программа поиска файлов;
- `директория` — каталог, в котором нужно искать. В нашем случае я указал системный **/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 больше, 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 дополнительное средство проверки. Наилучшим методом является подсчет контрольной суммы. Допустим, что вы хотите отслеживать изменения в директории **/etc**. Для этого выполните следующую команду:

```
md5sum /etc/*
```

Таким образом, подсчитывается контрольная сумма указанных в качестве параметра файлов. На экране вы получите результат выполнения команды примерно такого вида:

```
783fd8fc5250c439914e88d490090ae1  /etc/DIR_COLORS
e2eb98e82a51806fe310bffdd23ca851  /etc/Muttrc
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 данном случае указаны все файлы каталога **/etc/***. Результат расчета выводится на экран. Но запоминать эти данные неудобно, поэтому логично будет записать их в файл, чтобы потом использовать его содержимое для анализа изменений. Следующая команда сохраняет результат в файле **/home /flenov/md**:

```
md5sum /etc/* >> /home/flenov/md
```

Чтобы сравнить текущее состояние файлов директории **/etc** с содержимым файла **/home/flenov/md**, необходимо выполнить команду:

```
md5sum -c /home/flenov/md
```

На экране появится список всех файлов, и напротив каждого должна быть надпись "Success" (Успех). Это означает, что изменений не было. Давайте модифицируем какой-нибудь файл, выполнив, например, следующую команду:

```
groupadd test
```

Пока не будем вдаваться в подробности команды, сейчас достаточно знать, что она изменяет файл **/etc/group**. Снова выполняем команду проверки контрольных сумм файлов:

```
md5sum -c /home/flenov/md
```

Теперь напротив файла **/etc/group** будет сообщение об ошибке, т. е. контрольная сумма изменилась. Таким образом, даже если дата корректировки файла осталась прежней, по контрольной сумме легко определить наличие вмешательства.

Что контролировать

Некоторые администраторы следят только за файлами настройки. Это большая ошибка, потому что атакой хакеров может быть не только конфигурация,

но и исполняемые файлы. То, что Linux является продуктом с открытым кодом, имеет свои преимущества и недостатки.

Порок в том, что профессиональные хакеры знают программирование. Им не составляет труда взять исходный код какой-либо утилиты и изменить его на свое усмотрение, добавив необходимые функции. Таким образом, очень часто в системе открыты потайные двери.

Вы должны контролировать изменения как конфигурационных файлов, так и всех системных программ и библиотек. Я рекомендую следить за каталогами **/etc**, **/bin**, **/sbin** и **/lib**.

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 "0", который определяет конец имени файла. Все остальное можно применять.

Самое неприятное — это возможность использовать невидимые символы, т. к. хакер может создать программу, у которой в имени только нечитаемые знаки, и пользователь не видит такого файла. Таким образом, взломщики скрывают в ОС свои творения.

Рассмотрим пример с использованием перевода строки. Допустим, что хакер назвал свой файл hacker\nhosts.allow. В данном случае под "\n" подразумевается перевод каретки, а значит, имя 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, и администратор не заподозрит ничего страшного в таком имени.

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 текущую директорию. Администратор, выполнив команду `ls`, может не заметить, что существуют два файла с одинаковыми именами, а пробела все равно не видно.

Пробелы можно вставлять в любые имена файлов, например, перед именем (" hosts.allow") или наоборот, добавить в конец, и невнимательный администратор ничего не заметит. Чтобы увидеть конечный пробел, можно при выводе добавлять к каждому имени символ "/". Для этого при вызове команды `ls` используйте ключ `-F`.

Еще один 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 обнаружить что-либо невозможно, но если приглядеться повнимательнее, то вы увидите, что вместо букв l (L) стоит цифра 1 (единица).

Хакеры могут использовать этот прием. Еще можно подменять букву "b" на "d". И здесь трудно что-нибудь заподозрить, потому что если человек каждый день видит одно и то же, то, чаще всего, воспринимает желаемый текст за действительный.

Внимание — главное оружие администраторов. Вы должны проявлять интерес к любой мелочи, и нельзя позволить обмануть наше зрение.

3.1.3. Ссылки

В вашей системе могут появиться документы для совместного использования. Рассмотрим эту ситуацию на примере. Допустим, что файл отчетности **/home/report** должен быть доступен нескольким пользователям. Было бы логично, если копия этого файла находилась бы в домашних директориях этих пользователей. Но создавать несколько копий неудобно, потому что затруднится синхронизация. Да и сложно собрать в одно целое модификации из нескольких файлов, особенно если корректировался один и тот же кусок. Кто будет оценивать, чьи изменения необходимо вносить в общий файл?

Проблема решается с помощью ссылок, которые бывают жесткими (Hard link) и символьными (Symbolic link). Для постижения самой сути ссылок необходимо понимать, что такое файл и какое место ему отводится операционной системой. При создании файла на диске выделяется пространство для хранения данных. Его имя — это всего лишь ссылка из директории на участок диска, где физически находится файл. Получается, что можно создать несколько ссылок на одни и те же данные, и ОС Linux позволяет делать это.

Когда мы выполняем команду `ls -l`, то на экране появляется подробная информация о файлах в текущей директории. Напомню ее вид:

```
-rw-r--r--    1 Flenov   FlenovG     118 Nov 26 16:10 1.txt
```

Если к директиве добавить ключ `i` (выполнить команду `ls -il`), то к выводимой информации добавится еще и дескриптор файла:

```
913021 -rw-r--r--    1 Flenov   FlenovG     118 Nov 26 16:10 1.txt
```

Первое число и есть дескриптор, по которому определяется физическое расположение файла.

Жесткая ссылка указывает непосредственно на данные и имеет такой же дескриптор. Таким образом, файл физически не удаляется из системы, пока не будут уничтожены все жесткие ссылки. По сути, каждое имя файла уже является жесткой ссылкой на данные.

Для создания таких ссылок используется команда ln, которая имеет следующий вид:

```
ln имя_файла имя_ссылки
```

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 проверять все, что будет рассматриваться дальше, создайте в своей системе файл 1.txt. Для этого можно выполнить команду:

```
cat > 1.txt
```

Нажмите клавишу <Enter> и введите несколько строк текста и нажмите клавиши <Ctr>+<D>. Теперь у вас есть необходимый файл для тестирования.

Создадим 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 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 находится дескриптор 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 (список содержимого директории), а вторая — отображает документ 1.txt. Убедитесь, что содержимое обоих файлов изменилось и имеет одинаковые данные.

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 в третьей колонке уменьшилось значение счетчика ссылок до единицы.

Символьная ссылка указывает не на данные, а на имя файла. Это дает некоторые преимущества, но одновременно возникает большое количество проблем. Для создания символьной ссылки нужно использовать команду `ln` с ключом `-s`. Например:

```
ln -s link.txt symbol.txt
```

Посмотрим на результат с помощью команды `ls -il`:

```
913021 -rw-r--r-- 1 root root 519 Feb 22 12:19 link.txt
913193 lrwxrwxrwx 1 root root 8 Feb 22 12:40 symbol.txt -> link.txt
```

Теперь дескрипторы файлов разные, но для `link.txt` первый символ следующей колонки равен букве "`l`". Как раз она и указывает на то, что мы имеем дело с символьной ссылкой. Третий параметр равен единице, а последний — после знака "`->`" содержит имя файла, на который указывает ссылка.

Попробуем удалить основной файл и после этого просмотреть содержимое ссылки symbol.txt:

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

В первой строке мы удаляем файл link.txt. Вторая команда отображает список директорий. Убедитесь, что файла link.txt нет. Если вы используете Red Hat-дистрибутив, то команда `ls`, скорей всего, имеет псевдоним, который позволяет в зависимости от типа файла отображать его различными цветами. Если нет, то замените вторую команду на `ls --color=tty -il`.

Строка, содержащая информацию о ссылке symbol.txt, должна быть красного цвета, а текст — мигающий белый. Это говорит о том, что ссылка "битая", т. е. указывает на несуществующий файл. Команда `cat symbol.txt` пытается отобразить содержимое ссылки. Так как файла нет, мы увидим сообщение об ошибке.

Самое интересное, что если попытаться записать какие-либо данные в файл symbol.txt, то файл link.txt будет автоматически создан. Это огромный недостаток, поэтому вы должны следить за символьными ссылками перед удалением файлов.

Второй недостаток символьных ссылок кроется в правах доступа, но мы их будем рассматривать в *гл. 4*.

Еще один минус таится в блокировках. Если открыть на редактирование файл, для которого создана символьная или жесткая ссылка, то он блокируется. Представим себе, что существует ссылка на файл **/etc/passwd** или **/etc/shadow**. При блокировке одного из них вход в систему станет невозможным.

Чтобы взломщик не смог воспользоваться блокировками, его права на запись в системные каталоги должны быть ограничены. А пользователю в большинстве случаев надо давать разрешение писать только в свою домашнюю директорию и каталог **/tmp**. Иногда при разделении файлов может потребоваться работа с чужими каталогами, но все равно доступ ограничивается каталогом **/home**, где расположены пользовательские директории.

Глядя на все недостатки ссылок, возникает вопрос — а нужно ли действительно использовать их? Я рекомендую это делать только в крайнем случае, когда все остальные способы решения проблемы еще хуже. Но если нет другого выхода, то делайте это аккуратно.

3.2. Загрузка системы

Некоторые администраторы не обращают внимания на то, как стартует система. Для них главное — только работа ОС. Да, прямой зависимости нет. Но во время загрузки ОС запускается множество программ, которые отнимают память, уменьшая тем самым производительность системы.

Помимо этого, быстрая загрузка позволяет оперативно восстановить работу компьютера после сбоя. Все машины когда-либо приходится перезагружать, чтобы возобновить полноценное функционирование. Это происходит из-за ошибок в программном обеспечении, перебоев с электропитанием и др. Чем скорее вы сможете это сделать, тем меньше будет простой.

Во время загрузки должны производиться все необходимые настройки, чтобы сразу после старта не приходилось что-то 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

Для начала вернемся к утилите setup. Запустите ее в окне терминала, и перед вами откроется окно, как на рис. 2.12. Зайдите в раздел **System Services**. Здесь перечислены все установленные сервисы, а напротив тех, что запускаются автоматически, в квадратных скобках будет стоять звездочка. Если вы устанавливали какой-либо демон, который вам необходим в работе, но использовать его будете изредка, то нет смысла запускать его автоматически и открывать ворота для хакера. Лучше убрать для него флаг автозапуска и

стартовать только при необходимости, а сразу после работы останавливать сервис.

Например, я иногда отлаживаю на своем сервере Web-сценарии, требующие MySQL. Держать базу данных постоянно загруженной — расточительство памяти и лишняя дверь в систему. Поэтому я запускаю MySQL вручную по мере необходимости, и по окончании отладки прекращаю его работу.

Настоятельно рекомендую поступить так же и убрать все лишнее с глаз долой. Для этого стрелками выделите нужный демон и снимите галочку нажатием клавиши пробел. После того как вы настроили автозапуск, переключитесь клавишей <Tab> на пункт **OK**, чтобы сохранить изменения. Конечно, уже запущенные демоны не выгрузятся из памяти, но при следующем старте загружаться не будут. Перезагрузите компьютер и убедитесь в том, что система работает верно, и запускаются только необходимые демоны.

Если вы работаете в KDE или GNOME, то можно воспользоваться графической утилитой для настройки автоматически запускаемых демонов. Для этого на рабочем столе щелкните по значку **Control Panel** (Панель управления) и перед вами откроется окно, содержащее ссылки на основные программы конфигурирования системы. Нас будет интересовать ярлык **Службы**.

Эту же утилиту можно запустить и другим способом. Выберите главное меню ОС, а в нем пункт **Система** и, наконец, **Службы** (рис. 3.2). В дальнейшем, для обозначения программ, которые нужно запустить из главного меню, я буду просто писать:

Основное меню ОС/Система/Службы.

Главное окно графической утилиты Службы представлено на рис. 3.3. В центре формы вы можете увидеть список, состоящий из двух колонок. В первом столбце располагаются флажки, а во втором — названия демонов. Если напротив сервиса стоит галочка, то служба запускается автоматически. Щелчком по флажку вы можете устанавливать и снимать галочку.

Выделив какую-либо службу, вы можете ее запустить, остановить или перезапустить с помощью соответствующих кнопок на панели окна или через меню **Действия**. Любые изменения состояния автозагрузки необходимо сохранить. Для этого нажмите кнопку **Сохранить** на панели или выберите меню **Файл/Сохранить изменения**.

Внимание!

Никогда не запускайте службы, которыми вы не пользуетесь. В автозапуске должны находиться только те программы, которые необходимы вам или пользователям сервера регулярно. Если какой-либо демон используется редко, то не следует его устанавливать в автозапуск. Такие сервисы надо запускать только по мере надобности и останавливать сразу после применения. Ненужные службы лучше удалить совсем, чтобы не было соблазна использовать.

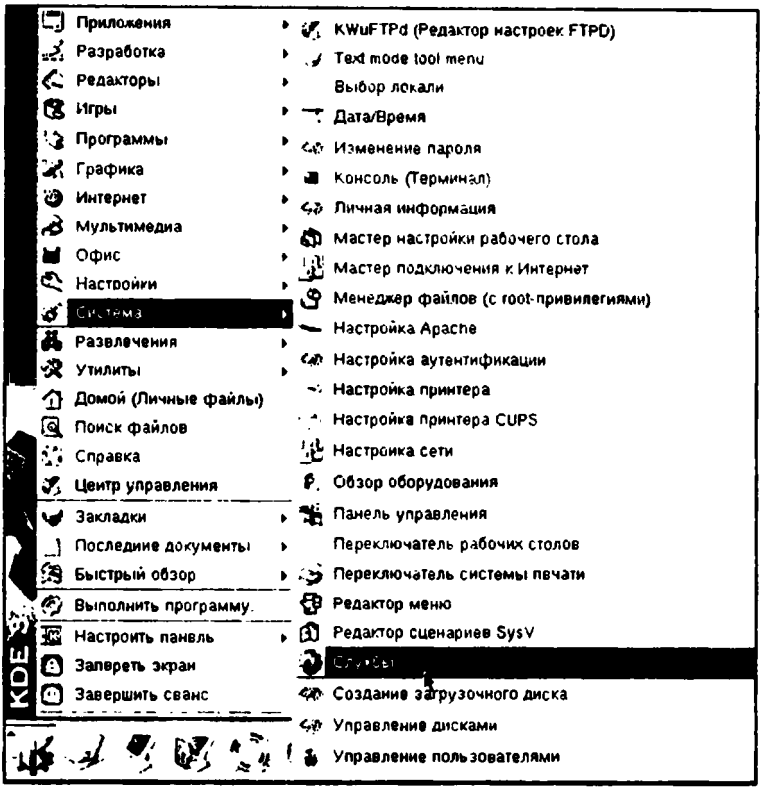

Рис. 3.2. Запуск утилиты Службы

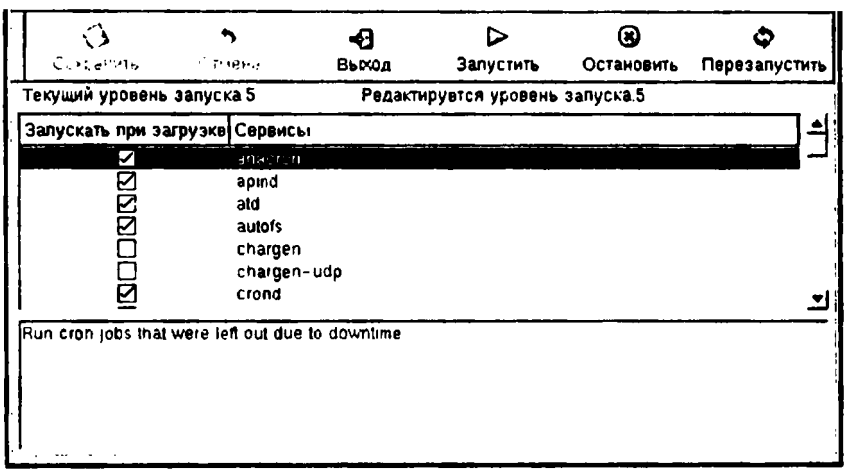

Рис. 3.3. Главное окно утилиты Службы

3.2.2. LILO

Как мы уже знаем, программа LILO позволяет загружать Linux и другие ОС, установленные на вашем компьютере. Все настройки загрузки хранятся в файле **/etc/lilo.conf**. Программа LILO начинает работать после того, как компьютер протестировал систему, и до старта ОС. В старых версиях это выглядело, как простое текстовое приглашение:

```
LILO
```

или

```
LILO boot:
```

В ответ на это вы можете нажать клавишу <Enter>, чтобы загрузить ОС по умолчанию или клавишами <↑> и <↓> выбрать ту ОС, которая нужна в данный момент. В современных версиях загрузчик имеет более приятный графический вид.

Пример конфигурационного файла можно увидеть в листинге 3.1.

Листинг 3.1. Конфигурационный файл lilo.conf

```
disk=/dev/hda
bios=128

boot=/dev/hda
prompt
timeout=300
lba32
default=linux-2.4.18

image=/boot/vmlinuz-2.4.18-5asp
initrd=/boot/initrd.2.4.18-5asp.img
label=linux-2.4.18
root=/dev/hda2
read-only
```

Каждая строка устанавливает какой-либо параметр загрузки. Таких опций много, и подробно с ними вы можете познакомиться в документации, поставляемой с ОС, а мы рассмотрим только основные. Большинству параметров присваивают значения. Это выглядит так:

```
Параметр=Значение
```

Слева от знака равенства находится имя параметра, а справа — значение, которое надо установить.

Рассмотрим возможные параметры файла lilo.conf:

- `boot=/dev/hda` — задает устройство, с которого происходит загрузка;
- `map=/boot/имя файла` — карта загрузки. Если этот параметр опущен, то по умолчанию будет использоваться файл **/boot/map**;
- `timeout=300` — время ожидания в миллисекундах. Если за этот промежуток времени ничего не выбрано, то будет загружена ОС по умолчанию;
- `lba32` — позволяет использовать возможности lba32 (32-разрядная адресация блоков диска). Присутствие этого параметра может вызвать проблемы со старыми жесткими дисками без поддержки LBA (Logical Block Addressing, логическая адресация блоков);
- `default=linux-2.4.18` — определяет, что будет загружаться по умолчанию. В данном случае выбран образ `linux-2.4.18`;
- `image=/boot/vmlinuz-2.4.18-5asp` — указывает на ядро Linux. Чаще всего эта строка выглядит как `/boot/vmlinuz`;
- `label=linux-2.4.18` — метка или текст, который будет появляться в окне загрузчика;
- `root=/dev/hda2` — диск, на котором расположена корневая файловая система;
- `read-only`.

На данный момент этой информации нам хватит. Большинство из нерассмотренных опций уже устарели и могут понадобиться только при использовании очень старых компьютеров (такие мало у кого сохранились). По своему опыту могу сказать, что перечисленных параметров будет достаточно. При компиляции ядра мы отредактируем lilo.conf и дадим возможность загрузки по выбору.

Для корректировки файла достаточно открыть его в любом текстовом редакторе. Тут серьезных изменений не бывает, поэтому я использую редактор, встроенный в программу MC. Для этого запустите Midnight Commander. Скорей всего, текущим каталогом будет ваша папка, например **/home/root**. Нужно перейти в корневой каталог. Выберите самую первую папку **/..** и нажмите клавишу <Enter>. Теперь вы окажетесь в папке **/home**. Повторите эту операцию и переместитесь в корневой каталог.

Теперь находим папку **/etc** и нажимаем клавишу <Enter>, чтобы войти в нее. Здесь ищем lilo.conf, выделяем его и нажимаем клавишу <F4>. Перед вами откроется простой текстовый редактор. Поменяйте нужные параметры. Нажмите клавишу <Esc> для выхода. Если данные были изменены, то программа выдаст запрос на сохранение. Соглашайтесь, если нужно принять исправления.

Если вы никогда не редактировали файлы, то попробуйте сделать это сейчас. Для примера можно изменить время ожидания выбора пользователем способа загрузки Linux: с жесткого диска или дискеты. Конечно же, с винчестера я загружаюсь намного чаще, включаю компьютер и ухожу готовить кофе. В дальнейшем мы не будем останавливаться на этом процессе так подробно.

В конфигурационных файлах Linux есть понятие комментария. Это текст, который программа просто игнорирует, и вы можете писать в нем любые пояснения или временно отключать какие-то параметры. Комментарий начинается с символа "#". Всем, что находится после этого символа, программа пренебрегает. Например:

```
# Это комментарий
boot=/dev/hda
timeout=300 # Это комментарий, показывает время загрузки
# lba32
default=linux-2.4.18 # Это комментарий, ОС по умолчанию
```

В этом примере первая строка начинается с символа "#", и она вся будет проигнорирована. В третьей строке комментарий стоит после указания параметра. Это значит, что сам параметр будет прочитан, а текст пояснения после него будет пропущен.

В четвертой строке перед lba32 стоит знак "#", значит этот параметр будет проигнорирован и воспринят как комментарий.

Пояснения очень удобны для того, чтобы временно отключать какие-либо опции. Вы можете просто удалить параметр и забыть, как он был написан, но если вы превратили его в комментарий, то для возврата достаточно убрать знак "#", и параметр заработает.

С помощью утилиты LILO можно защитить ваш компьютер от несанкционированной загрузки ОС. Это необходимо, потому что при старте системы можно выполнить команду уже на этапе загрузки. Если злоумышленник получил доступ к вашему компьютеру, то легко может войти в однопользовательском режиме с последующим взломом пароля root или реализовать команду.

Если при загрузке LILO отображается в виде простого текстового приглашения (характерно для дистрибутивов Red Hat), то необходимо ввести имя загружаемой системы (linux), потом ключевое слово `init=` и команду:

```
Linux Boot: linux init=команда
```

Чтобы хакер не смог запустить систему, необходимо защитить LILO с помощью пароля. Для этого в конфигурационном файле после ключевого слова `image` добавьте строку:

```
password=пароль
```

Вот пример задания пароля qwerty на загрузку:

```
image=/boot/vmlinuz-2.4.18-5asp
password=qwerty
initrd=/boot/initrd.2.4.18-5asp.img
label=linux-2.4.18
root=/dev/hda2
read-only
```

Если на вашем компьютере установлены разные варианты системы, то необходимо для каждого из них указывать пароль. В следующем примере LILO позволяет загружать два ядра, и для обоих вводится свой пароль:

```
image=/boot/vmlinuz-2.4.18-5asp
password=qwerty
initrd=/boot/initrd.2.4.18-5asp.img
label=linux-2.4.18
root=/dev/hda2
read-only

image=/boot/vmlinuz-2.6.2
password=123456
initrd=/boot/initrd.2.6.2.img
label=linux-2.6.2
root=/dev/hda2
read-only
```

О конфигурировании LILO с двумя ядрами мы поговорим в *разд. 3.8.4.*

Если добавить параметр `password` до описания `image`, то указанный пароль будет действовать для всех ОС и ядер, загружаемых с помощью LILO.

Но пароль запрещает только основную загрузку, а возможность выполнения команд при старте системы сохраняется. Чтобы сделать и это невозможным, добавьте в конфигурационный файл lilo.conf после объявления пароля строку с ключевым словом `restricted`.:

```
image=/boot/vmlinuz-2.4.18-5asp
password=qwerty
restricted
initrd=/boot/initrd.2.4.18-5asp.img
label=linux-2.4.18
root=/dev/hda2
read-only
```

Чтобы внесенные в файл изменения вступили в силу, необходимо запустить в командной строке директиву `lilo`. Таким образом, новые параметры будут

записаны в загрузочную область и начнут действовать при следующем старте системы.

3.2.3. init

С помощью LILO запускается программа загрузки ОС, которая настраивает все необходимое оборудование, загружает драйверы и монтирует жесткие диски. По окончании этого процесса с винчестера запускается программа init, которая завершает загрузку.

У программы init, как и у большинства других утилит Linux, есть свой конфигурационный файл, в котором можно производить определенные настройки (листинг 3.2). Этот файл называется inittab и расположен в папке **/etc** (полный путь **/etc/inittab**).

Листинг 3.2. Файл настройки программы inittab

```
#
# inittab       This file describes how the INIT process
#               should set up
#               the system in a certain run-level.
#
# Author:       Miquel van Smoorenburg,
#               <miquels@drinkel.nl.mugnet.org>
#               Modified for RHS Linux by Marc Ewing and
#               Donnie Barnes
#

# Default runlevel. The runlevels used by RHS are:
#   0 - halt (Do NOT set initdefault to this)
#   1 - Single user mode
#   2 - Multiuser, without NFS (The same as 3, if you do not
#   have networking)
#   3 - Full multiuser mode
#   4 - unused
#   5 - X11
#   6 - reboot (Do NOT set initdefault to this)
#  id:5:initdefault:

# System initialization.
si::sysinit:/etc/rc.d/rc.sysinit

# What to do in single-user mode.
~~:S:wait:/sbin/sulogin
```

```
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6

# Things to run in every runlevel.
ud::once:/sbin/update

# Trap CTRL-ALT-DELETE
ca::ctrlaltdel:/sbin/shutdown -t3 -r now

# When our UPS tells us power has failed,
#  assume we have a few minutes
# of power left.  Schedule a shutdown for 2 minutes from now.
# This does, of course, assume you have powerd installed
# and your
# UPS connected and working correctly.
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"

# If power was restored before the shutdown kicked in,
# cancel it.
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"

# Run gettys in standard runlevels
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6

# Run xdm in runlevel 5
# xdm is now a separate service
x:5:respawn:/etc/X11/prefdm -nodaemon
```

Файл начинается с комментария, в котором дана информация о модуле и авторе, затем — описание различных уровней, которые поддерживаются системой. В некоторых дистрибутивах их может быть от двух, а Red Hat Linux-подобные системы поддерживают целых семь. Рассмотрим каждый уровень:

- 0 — остановка ОС;
- 1 — текстовый однопользовательский режим, используется редко и только администраторами для выполнения критически важных изменений;

- 2 — текстовый многопользовательский, локальный режим (нет поддержки сети);
- 3 — текстовый многопользовательский, сетевой режим;
- 4 — не используется, но в дальнейшем может быть задействован;
- 5 — графический режим;
- 6 — перезагрузка системы.

Пока что мы рассмотрели семь уровней. Есть еще уровень S, что соответствует однопользовательскому режиму, он применяется в файлах сценариев, но иногда присутствует и в inittab.

Теперь познакомимся со структурой файла. Каждая строка в нем (не считая комментариев и пустых строк) выглядит следующим образом:

```
идентификатор:уровни:флаги:действие
```

Строка состоит на четырех аргументов, отделенных между собой двоеточием. Давайте рассмотрим каждый параметр в отдельности:

- `идентификатор` — уникальный номер строки, который имеет произвольное значение. Единственное ограничение — в файле не должно быть двух строк с одинаковыми идентификаторами;
- `уровни` — режимы, в которых будет выполняться команда. Например, если ей надлежит работать на втором и третьем уровнях, то здесь должно стоять число 23. Это просто цифры 2 и 3, которые не разделяются пробелами или какими-либо другими знаками. Если команду нужно выполнять на любом уровне, то этот параметр должен быть пустым;
- `флаги` — задает поведение команды, может принимать одно из следующих значений:
  - `boot` — выполняется только один раз при загрузке ОС. В этом случае параметр `уровни` просто игнорируется;
  - `bootwait` — равносильно указанию параметров `boot` и `wait` одновременно, т. е. выполнение должно быть во время загрузки ОС, и при этом нужно дождаться завершения операции;
  - `ctrlaltdel` — нажата комбинация клавиш <Ctrl>+<Alt>+<Del>. Случайных или незапланированных перезагрузок быть не должно. Наличие трех клавиш — это недостаток, потому что любой хакер может подойти и нажать их ради шутки или в корыстных целях. Я рекомендую отключить эту возможность, установив в начале строки знак комментария "#";
  - `initdefault` — строка с этим параметром читается только при первом обращении к init и определяет уровень загрузки. Если в этой строке бу-

дет стоять число 5, то ОС будет сразу в графический режим. Если вам нужно загрузить ОС в текстовый, многопользовательский режим с поддержкой сети, то измените значение параметра на цифру 3 (см. описание уровней загрузки). Можно указать несколько чисел (я не рекомендую этого делать), в этом случае будет выбрано максимальное;

- `off` — отключить выполнение команды, что равноценно превращению строки в комментарий или даже удалению. Но если этот процесс уже работает, то ему передается сигнал, требующий завершения программы;
- `once` — выполнять команду только один раз;
- `powerfail` — отключение электроэнергии. Ошибку можно увидеть только при наличии источника бесперебойного питания, подключенного к компьютеру через специальный интерфейс (чаще всего через COM-порт);
- `powerokwait` — подача электроэнергии возобновилась;
- `powerwait` — система перешла в режим ожидания восстановления питания. Предполагается, что к компьютеру подключен источник бесперебойного питания, который сообщил об ошибке (отсутствии питания);
- `respawn` — если во время выполнения произойдет ошибка, то повторить команду. Это бывает необходимо для критически важных программ, которые должны быть всегда в запущенном состоянии;
- `sysinit` — такие команды выполняются перед тем, как появится консоль (приглашение ввести пароль). При этом ОС дожидается окончания выполнения команды;
- `wait` — ожидать завершения процесса, а значит, программа init не будет выполнять других действий, пока не закончится выполнение команды, указанной в этой строке. Например, если нужно проверить состояние диска, то необходимо заморозить дальнейшую загрузку до окончания проверки;

❒ `Действие` — команда, которая должна быть выполнена.

Теперь рассмотрим некоторые строки из листинга 3.2, чтобы понять их смысл и научиться управлять ими для повышения эффективности системы.

Самая первая строка после комментариев в нашем файле выглядит так:

```
id:5:initdefault:
```

Мы уже знаем, что параметр `initdefault` определяет, как будет загружаться система. В данном случае файл init будет выполняться с 5 уровнем, т. е. в графическом режиме.

Следующим интересным моментом являются строки:

```
l0:0:wait:/etc/rc.d/rc 0
l1:1:wait:/etc/rc.d/rc 1
l2:2:wait:/etc/rc.d/rc 2
l3:3:wait:/etc/rc.d/rc 3
l4:4:wait:/etc/rc.d/rc 4
l5:5:wait:/etc/rc.d/rc 5
l6:6:wait:/etc/rc.d/rc 6
```

Как видите, здесь для каждого уровня своя строка, и при этом запускается один и тот же файл **/etc/rc.d/rc**, но после имени через пробел стоит число, соответствующее текущему уровню. Это параметр, который увидит программа **/etc/rc.d/rc**.

Что это за программа? Основная ее задача — убить все текущие процессы и запустить другие, соответствующие новому режиму выполнения. Например, вы работали на 3 уровне (полный многопользовательский режим) и хотите перейти на 1 (однопользовательский). Как это происходит? Все программы многопользовательского режима завершаются, а потом активизируются только те процессы, которые соответствуют однопользовательскому режиму, и ничего лишнего в памяти не должно быть. Именно это и делает программа **/etc/rc.d/rc**.

Перейдите в папку **/etc/rc.d/** и посмотрите на ее содержимое. Помимо программы, здесь есть каталоги с именами `rcX.d`, где `X` — это число от 0 до 6, соответствующее уровню выполнения. В каждой папке есть файлы, имена которых начинаются с буквы "K" или "S". При выходе с уровня выполняются все файлы первого типа, они уничтожают все запущенные на нем процессы. А при входе на уровень выполняются файлы на букву "S", которые активизируют все необходимые процессы данного уровня.

Таким образом, система может обезопасить себя тем, что на одном уровне будут работать только те программы, которые должны быть в однопользовательском режиме, и многопользовательское вторжение становится невозможным при правильном конфигурировании скриптов, запускающих и останавливающих процессы. Конечно же, вручную править скрипты не приходится, и система следит за файлами без нас, но знать о такой особенности загрузки вы должны. Например, вы хотите, чтобы какой-то демон не запускался при старте системы на третьем уровне. Для этого можно просто удалить соответствующий файл из каталога **/etc/rc.d/rc3.d**, или сделать так, чтобы его имя не начиналось с букв "S" или "K".

В каталоге **/etc/rc.d/init.d** находятся скрипты, которые запускают, останавливают или перезапускают сервисы в системе. Именно эти файлы используются при переходе с одного уровня на другой.

Очень интересной является строка:

```
ca::ctrlaltdel:/sbin/shutdown -t3 -r now
```

Она выполняется на любом уровне, потому что второй аргумент отсутствует. Третий параметр равен `ctrlaltdel`. Это значит, что по нажатию клавиш <Ctrl>+<Alt>+<Del> будет выполнена указанная команда. В большинстве ОС такая комбинация используется для перезагрузки системы. Какая директива выполнится в Linux? Это `/sbin/shutdown -t3 -r now`. Мы уже знаем, что команда `shutdown` с ключом `-r` — это перезагрузка. Параметр `-tX` задает время задержки, где `X` — количество секунд до рестарта.

Получается, что по нажатию клавиш <Ctr>+<Alt>+<Del> запустится директива, требующая через три секунды перезагрузить систему.

Теперь посмотрим на строку, которая будет выполнена при сбое питания:

```
pf::powerfail:/sbin/shutdown -f -h +2 "Power Failure; System Shutting Down"
```

Строка выполняется на любом уровне. Здесь снова участвует команда `shutdown`, но теперь с другими ключами:

- `-f` — отменяет проверку диска fsck;
- `-h` — указывает на необходимость выключения питания;
- `+2` — задает время в минутах до перезагрузки;
- далее идет в кавычках сообщение, которое появится на каждой консоли. В графическом режиме все пользователи увидят окно с предупреждением такого же содержания.

Обратите внимание, что время до перезагрузки 2 минуты. Это очень мало, т. к. простые источники бесперебойного питания могут продлить работу компьютера до 20 минут. А если учесть, что серверы чаще всего стоят без монитора или, в крайнем случае, с выключенным дисплеем, то жизнь сервера может быть продлена до 40 минут. Значение в две минуты явно занижено, и я рекомендую проверить документацию на ваш источник бесперебойного питания и увеличить тайм-аут, чтобы сервер не ушел в перезагрузку раньше времени.

Теперь посмотрим, что происходит при восстановлении питания:

```
pr:12345:powerokwait:/sbin/shutdown -c "Power Restored; Shutdown Cancelled"
```

Команда выполняется на уровнях с 1 по 5 и отменяет перезагрузку. На 0 и 6 уровнях, когда уже начался процесс выключения системы или рестарт, аннулировать что-либо уже поздно.

Отмена происходит вызовом команды `shutdown` с ключом `-c`, после чего идет текст сообщения о том, что питание восстановилось, и можно работать.

Теперь посмотрим на следующие строки:

```
1:2345:respawn:/sbin/mingetty tty1
2:2345:respawn:/sbin/mingetty tty2
3:2345:respawn:/sbin/mingetty tty3
4:2345:respawn:/sbin/mingetty tty4
5:2345:respawn:/sbin/mingetty tty5
6:2345:respawn:/sbin/mingetty tty6
```

Здесь на шести терминалах (ttyX, где X — это номер терминала или виртуальной консоли) выдается сообщение на вход в систему. Это не есть хорошо, потому что хакер может воспользоваться любой консолью для проникновения в систему (в качестве администратора). Чтобы этого не произошло, можно запретить вход со всех терминалов, кроме одного. Для этого нужно заменить флаг `respawn` на `off`.

Вообще-то можно это и не делать. Есть способ лучше — конфигурирование настроек tty, которые находятся в файле **/etc/securetty**. Пример такого файла вы можете увидеть в листинге 3.3.

Листинг 3.3. Файл /etc/securetty

```
vc/1
vc/2
vc/3
vc/4
vc/5
vc/6
vc/7
vc/8
vc/9
vc/10
vc/11
tty1
tty2
tty3
tty4
tty5
tty6
tty7
tty8
tty9
tty10
tty11
```

Этот файл определяет консоли и терминалы, с которых можно подключаться с правами root. Первые одиннадцать строк определяют 11 виртуальных консолей, остальные — назначают окна терминалов. Чтобы разрешить вход с одного терминала, оставьте только строку tty1, а все остальные удалите. Таким образом, терминалы будут доступны для работы, но только с первого можно подключиться с правами root.

Для переключения между виртуальными консолями нужно нажать клавишу <Alt> и любую клавишу от <F1> до <F6>.

Самая последняя строка выполняется только на 5 уровне:

```
x:5:respawn:/etc/X11/prefdm -nodaemon
```

Эта строка запускает команду **/etc/X11/prefdm**, которая переводит работу в графический режим и отображает соответствующее окно входа в систему, которое мы рассматривали в *разд. 2.7*. Если вы используете текстовый режим, то для запуска графической оболочки можно использовать описанную ранее команду.

Если нужно, чтобы программа инициализации просмотрела конфигурационный файл inittab без смены уровня, то можно вручную принудить init запуститься. Для этого выполните команду:

```
/etc/init q
```

Для перехода на другой уровень необходимо выполнить команду:

```
telinit X
```

где X — новый уровень, на котором должна работать ОС. Эта команда удобна, если вы грузитесь на пятом уровне. В этом случае легко перейти в текстовый режим на уровень 3. Если завершать графический режим штатными средствами оболочки, то ОС не будет выходить в текстовый режим, а останется графическое приглашение на ввод пароля.

А теперь небольшой фокус. Что будет, если воспользоваться командой telinit 6? Конечно же, начнется перезагрузка системы (в соответствии с назначением 6 уровня). А при выполнении команды telinit 0 (переход на нулевой уровень) произойдет выключение системы, как при выполнении команды shutdown -h now.

И все же я не советую употреблять переходы на 0 и 6 уровни, а использовать законный выход, т. е. команду shutdown.

3.2.4. Интересные настройки загрузки

Рассмотрим парочку файлов, которые хоть и незначительно, но влияют на загрузку.

Прежде чем появится приглашение ввести пароль, на экране отображается текстовая информация, пояснение. Чаще всего, здесь разработчик пишет имя дистрибутива и его версию. Эта информация хранится в файле **/etc/issue**, и вы легко можете его изменить в любом текстовом редакторе, в том числе и во встроенном в Midnight Commander.

После входа в систему тоже может выводиться текстовое сообщение, но по умолчанию в большинстве дистрибутивов оно отсутствует. Текст этого сообщения находится в файле **/etc/motd**, он может содержать новости для пользователей системы или каким-либо образом информировать об изменениях. Например, каждого первого числа месяца напоминать о необходимости сменить пароль.

3.3. Регистрация в системе

Теперь познакомимся с процессом регистрации пользователя в системе. Это поможет вам лучше понять систему безопасности, которая используется в ОС Linux при авторизации.

Мы уже знаем из *разд. 3.2*, что программа init 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/password**, а пароль — с соответствующей записью в файле **/etc/shadow**. Все пароли в файле хранятся только в зашифрованном виде. Для сопоставления введенный пароль тоже шифруется, и результат сравнивается со значением в файле **/etc/shadow** для указанного имени пользователя.

Почему так сложно происходит проверка? Просто все пароли в файле **/etc/shadow** зашифрованы необратимым алгоритмом (чаще всего используется алгоритм MD5). Это значит, что математическими методами из результата кодирования нельзя получить исходный пароль, поэтому возможен только подбор. Для этого существует несколько очень простых программ. Чем проще пароль и меньше его длина, тем быстрее программа найдет нужный вариант. Если пароль сложен и его длина более 8 символов, а лучше — свыше 16, то подбор может отнять слишком много времени.

Если идентификация пользователя состоялась, то программа login выполнит все автоматически загружаемые сценарии и запустит оболочку (командную строку), через которую и будет происходить 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 командную оболочку (Shell) невозможно, нам останется доступной лишь консоль getty, которая умеет только запрашивать имя пользователя и передавать его программе login.

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

3.3.1. Теневые пароли

В старых версиях Linux список пользователей и пароли хранились в файле **/etc/password**. Это не очень хорошо, потому что данный файл должен быть доступен для чтения всем пользователям, т. к. имена пользователей требуются очень многими безобидными программами. Например, при выполнении команды ls (просмотр файлов текущего каталога) нужно получить доступ к списку пользователей для получения имен владельцев файлов. Поскольку файл легко прочитать любому пользователю, то и зашифрованные варианты паролей тоже доступны, а значит, любой хакер сможет запустить подбор паролей и ждать заветного часа X, когда будет найдена нужная комбинация.

Чтобы защитить пароли, во всех современных версиях Linux их прячут в файл **/etc/shadow**, который доступен для чтения только администратору root. Файл **/etc/password** остался открытым для всех, но теперь в нем уже нет пароля. Давайте посмотрим, как выглядит файл **/etc/password**. Для примера я взял только верхние три строки из своего файла:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
```

Каждая строка содержит информацию о пользователе — семь аргументов, разделенных между собой двоеточием. Давайте разберем каждый из них:

- имя пользователя — login, который вы вводите;
- пароль — если вывод затенен, то вместо пароля будет стоять символ x;
- UID — уникальный идентификатор пользователя;
- GID — уникальный идентификатор группы;
- информация о пользователе — здесь может быть полное имя, адрес и т. д.;
- домашняя директория — каталог, принадлежащий пользователю, с которым он начинает работать при входе в систему;
- интерпретатор команд — оболочка, которая будет выполнять команды пользователя. Если интерпретатора команд не должно быть, то указывается файл **/sbin/nologin**.

Теперь посмотрим на строку для пользователя root. Первый параметр — это имя, и, конечно же, тут написано root. Пароля в файле нет, т. к. вместо него мы видим символ x (или !!), а он находится в соответствующей записи файла **/etc/shadow**.

Следующие два параметра — уникальный идентификатор пользователя (UID) и уникальный идентификатор группы (GID). В файле не может быть двух записей с одним и тем же UID. По GID система находит группу, в которую входит пользователь и, соответственно, определяет права, которые даны этой группе, а значит, и пользователю.

Информация о пользователе может быть любой, и на работу системы она не влияет. Это просто пояснение, которое администратор использует по своему усмотрению.

Далее идет домашняя директория. Это каталог, который открывается пользователю при входе в систему.

Последний параметр — это командный интерпретатор, который будет обрабатывать пользовательские запросы. Наиболее распространенным является интерпретатор **/bin/bash**. Если команды пользователя не должны выполняться, то в качестве этого параметра устанавливается **/sbin/nologin**. Именно это значение введено для записей **bin**, **daemon** и многих других, потому что под ними нельзя входить в систему и они предназначены только для внутреннего обеспечения безопасности определенных программ.

Теперь посмотрим на файл **/etc/shadow**, а точнее, возьмем только первые три строки. Для примера этого будет достаточно:

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

Здесь также несколько параметров, разделенных двоеточием. Нас будут интересовать первые два: login и пароль. По имени пользователя происходит связь записи из файла **/etc/shadow** с файлом **/etc/password**. А вот во втором параметре уже находится настоящая зашифрованная версия пароля. Но если вместо него стоят звездочки, то это запрет на использование записи. Например, для пользователей **bin** и **daemon** установлены именно звездочки, а значит, под этими учетными записями нельзя входить на компьютер.

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 ре-

дактированием, а если нет, то посмотрим, как получить доступ с загрузочной дискеты.

Загрузившись с дискеты, 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 директория **/mnt/restore** (желательно, чтобы она существовала 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 знаком двоеточия). В моем случае получилось:

```
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 поменять пароль, иначе это будет опасно 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 код. Такой подход исключает случайную опечатку при наборе и может с определенной долей вероятности гарантировать, что сохранен верный пароль.

3.3.3. Модули аутентификации

Аутентификация на основе двух файлов **/etc/passwd** и **/etc/shadow** немного устарела и предоставляет нам слишком скудные возможности. Разработчики ядра ОС Linux стараются исправить ситуацию с помощью добавления новых алгоритмов шифрования, но все эти попытки чисто косметические, а нам необходимо кардинальное изменение.

Абсолютно новое решение для реализации аутентификации предложила компания Sun — Pluggable Authentication Modules (PAM, подключаемые модули аутентификации).

Преимущество модульной аутентификации заключается в том, что не требуется перекомпиляция программы для их использования. Существуют модули для основных методов аутентификации, таких как Kerberos, SecureID, LDAP и др.

Конфигурационные файлы для каждого сервиса, который может использовать PAM, находятся в директории **/etc/pam.d**. В большинстве случаев вам не придется создавать эти файлы вручную, потому что они устанавливаются во

время инсталляции программы из RPM-пакета. Но вы должны знать их структуру, чтобы можно было изменить какие-то параметры в случае необходимости.

Каждая строка в конфигурационном файле состоит из четырех полей, разделенных пробелами:

- тип модуля — может принимать одно из следующих значений:
  - `auth` — аутентификация и проверка привилегий пользователей;
  - `account` — распределение ресурсов системы между пользователями;
  - `session` — поддержка сессии и регистрации действий пользователей;
  - `password` — проверка пароля;
- флаг — определяет параметры модуля. Здесь можно использовать три значения:
  - `required` — обязательный;
  - `optional` — необязательный;
  - `sufficient` — достаточный;
- полный путь к файлу модуля;
- аргументы модуля.

Рассмотрим пример конфигурационного файла для FTP-сервиса, который находится в файле **/etc/pam.d/ftp**.

```
#%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.

3.4. Процессы

Для того чтобы эффективно управлять своим компьютером, вы должны досконально изучить свой сервер и работающие на нем процессы. Взломав ваш сервер, злоумышленник постарается запустить на нем какую-либо программу, которая незаметно будет выдавать хакеру права root в системе. Таких программ в сети великое множество, и к ним относятся различные троянские программы.

Процесс — это программа или ее потомок. При запуске программы создается новый процесс, в котором и работает код. Каждая программа должна функционировать с определенными правами. Сервисы, которые активизируются при старте системы, обладают правами root или nobody (без прав). Программы, которые выполняются из командной строки, наделены правами запустившего пользователя, если не указан SUID- или SGID-бит, при котором программа имеет права владельца.

Существует два основных типа процессов — фоновый (background) и центральный (foreground). Центральный процесс для определенного терминала может быть только один. Например, запустив по команде `ls` программу man для просмотра помощи, вы не сможете выполнять другие команды, пока не выйдете из программы man.

У тех, кто знаком с программой Midnight Commander, может возникнуть вопрос — а как же тогда работает MC и одновременно в нем можно выполнять команды? Ответ прост, процессы могут порождать другие процессы. То же происходит и в MC, но если его закрыть, то закроются и все порожденные процессы.

3.4.1. Смена режима

Фоновыми процессами являются все сервисы. Они выполняют свои действия параллельно с вашей работой. Но вы в фоновом режиме можете запустить любую программу. Для этого достаточно после указания команды через пробел поставить знак "&". Например, выполните сейчас следующую команду:

```
man ls &
```

В ответ на это вы не увидите файл 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. Это число будет последовательно увеличиваться. В данном случае это первая команда, поэтому в квадратных скобках стоит единица. Это число формируется для каждого пользователя. Если войти в систему через второй терминал и запустить фоновый процесс, то вы увидите примерно следующее:

```
[1] 2805
```

В квадратных скобках опять число 1, а вот следующее значение отличается от выведенного на первом терминале и будет всегда другим. Это PID (Process ID, идентификатор процесса) созданного процесса, является уникальным для всех пользователей. Это значит, что если вы запустили процесс с номером 2802, то другой пользователь никогда не увидит этого идентификатора. Его PID будет другим.

Запомните идентификаторы, которые вы увидели, впоследствии они пригодятся.

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

```
[1] + Stopped      man ls
```

В данном случае 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 директиву, 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 клавиши <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.

3.4.2. Остановка процессов

Чтобы прекратить работающий процесс, необходимо сделать его центральным и остановить штатными средствами. Чаще всего, на экране есть подсказка, которая поможет выйти из программы. Если она отсутствует, то следует обратиться к документации или просмотреть файл помощи к программе через вызов `man имяпрограммы`.

Процессы, работающие только в фоне, не могут быть выведены на передний план. Для того чтобы их остановить, есть специализированные команды, которые чаще всего имеют вид:

```
имясервиса stop
```

Иногда процессы зависают. Да, такие ситуации бывают и в ОС Linux. Центральный процесс может быть снят с помощью комбинации клавиш <Ctrl>+<C> или <Ctrl>+<Break>. Но этот метод срабатывает не во всех случаях и не для всех программ. Если не удается завершить процесс по-хорошему, то можно поступить иначе. Для этого существует команда `kill`. Чтобы отключить процесс по личному идентификатору (тот, что мы видели в квадратных скобках), используйте команду:

```
kill %n
```

Параметр `n` нужно заменить на номер процесса. Например, чтобы завершить работу фоновой программы man, нужно выполнить:

```
kill %1
```

Затем сразу же запустите команду `jobs`. Вы должны увидеть на экране сообщение типа:

```
[1] + Terminated    man ls
```

После повторного вызова команды `jobs` программы man больше не будет.

Если вы хотите завершить работу процесса, который запущен не вами, но вы знаете его PID, то нужно выполнить команду:

```
kill n
```

Знак процента в этом случае не нужен. Тогда команда `kill` ищет процесс, у которого PID равен указанному числу `n` и посылает сигнал для завершения.

3.4.3. Просмотр процессов

С помощью команды `jobs` вы можете увидеть только запущенные вами процессы. Чтобы полюбопытствовать, чем занимаются остальные пользователи в системе, нужно выполнить команду `ps`. Если запустить ее без параметров, то результат на экране будет примерно следующий:

```
PID  TTY      TIME     CMD
1652 tty1     00:00:00 bash
1741 tty1     00:00:00 ps
```

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

Это далеко не полный список. Чтобы увидеть все процессы, следует выполнить команду `ps` с ключом `-a`. Но и это еще не весь перечень, потому что отобразятся только программы своего терминала. Если требуется полный список процессов, запущенных со всех терминалов, то нужно добавить ключ `-x`. По-

мимо 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 -axu
```

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 данный момент;
- `T` (Traced or Stopped) — в состоянии отладки или остановлен;
- `Z` (Zombied) — зависший. Такие 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 стоит вопросительный знак, то это означает, что процесс запущен еще на этапе загрузки системы и не принадлежит какому-либо терминалу.

Это 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 всех. Я люблю сохранять результат работы в текстовый файл, а потом спокойно изучать его в любом редакторе. Для этого я выполняю команду:

```
ps -axu >> ps.txt
```

Чтобы увидеть, чем в данный момент занимаются другие пользователи, можно выполнить команду w. В результате вы получите на экране приблизительно такую картину:

```
10:59am up 37 min, 2 users, load average: 0.00, 0.00, 0.00
USER   TTY   FROM       LOGIN@   IDLE   JCPU   PCPU WHAT
root   tty1  -         10:24am 0.00s 0.82s 0.05s w
flenov tty2  -          10:39am 8:13  0.85s 0.03s grotty
```

Из данного примера понятно, что в системе находятся два пользователя. На первом терминале работает пользователь root, а на втором — flenov. Очень удобно определять по списку, когда пользователь вошел в систему (колонка LOGIN@) и что делает в данный момент (колонка WHAT).

Посмотрите на столбцы JCPU и PCPU, по ним можно оценить загрузку системы. Если ваш компьютер начал работать слишком медленно, то можно увидеть, какой процесс отнимает много процессорного времени.

Команда ps выводит статическую информацию. Для наблюдения за нагрузкой системы удобнее использовать программу top. Она отображает список процессов, отсортированный по убыванию в зависимости от нагрузки (рис. 3.4). Таким образом, вы можете увидеть, какой сервис или программа отнимает драгоценные ресурсы и не дает нормально работать с компьютером.

```
 11:44am  up  1:22,  2 users,  load average: 0,00, 0,00, 0,00
58 processes: 46 sleeping, 1 running, 0 zombie, 11 stopped
CPU states:  0,0% user,  0,5% system,  0,0% nice, 99,4% idle
Mem:   255884K av,  240416K used,    7468K free,       0K shrd,   60188K buff
Swap:  514040K av,       0K used,  514040K free                   61472K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
 1847 root      16   0  1036 1036   824 R     0,3  0,4   0:00 top
  859 mysql     15   0  4380 4380  1644 S     0,1  1,6   0:00 mysqld
    1 root      15   0   452  452   400 S     0,0  0,1   0:04 init
    2 root      15   0     0    0     0 SW    0,0  0,0   0:00 keventd
    3 root      15   0     0    0     0 SW    0,0  0,0   0:00 kapmd
    4 root      34  19     0    0     0 SWN   0,0  0,0   0:00 ksoftirqd_CPU0
    5 root      15   0     0    0     0 SW    0,0  0,0   0:00 kswapd
    6 root      25   0     0    0     0 SW    0,0  0,0   0:00 bdflush
    7 root      15   0     0    0     0 SW    0,0  0,0   0:00 kupdated
    8 root      25   0     0    0     0 SW    0,0  0,0   0:00 mdrecoveryd
   17 root      15   0     0    0     0 SW    0,0  0,0   0:00 kjournald
  525 root      15   0   540  540   440 S     0,0  0,2   0:00 syslogd
  530 root      15   0   436  436   336 S     0,0  0,1   0:00 klogd
  551 rpc       15   0   540  540   456 S     0,0  0,2   0:00 portmap
  579 rpcuser   15   0   740  740   636 S     0,0  0,2   0:00 rpc.statd
  683 root      15   0   460  460   412 S     0,0  0,1   0:00 apmd
  737 ident     17   0   896  896   716 S     0,0  0,3   0:00 identd
  750 ident     15   0   896  896   716 S     0,0  0,3   0:00 identd
```

Рис. 3.4. Результат работы программы top

Если мой компьютер начинает "тормозить", или работа замедляется через определенные промежутки времени, то я запускаю top в отдельном терминале и по мере необходимости переключаюсь на него, чтобы увидеть нагрузку процессов.

Вверху окна выводится количество пользователей, общая загрузка системы и статистика процессов: общее количество, спящие, выполняемые, зависшие и остановленные.

Помимо этого, можно увидеть краткий отчет по использованию памяти: количество занятой и свободной оперативной памяти и размер раздела подкачки. В моем случае в компьютере установлено 256 Мбайт памяти, и из них свободно только 7 Мбайт, а раздел подкачки пока не используется. Такое малое количество свободной памяти говорит о том, что не помешало бы ее нарастить. Чем меньше компьютер использует swap-файл, тем лучше он работает. Конечно, пока что этот файл практически не используется, но если перейти в графический режим и запустить пару мощных приложений, то и Swap-раздела не хватит.

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

3.5. Планирование задач

Очень часто возникает необходимость выполнить какую-либо операцию в определенное время. Раньше я надеялся на свою память и вручную выполнял команды. Но когда несколько раз произошла осечка — просто был слишком занят, чтобы обратить внимание на часы и выполнить нужные действия, — я переложил задачу по слежению за временем на компьютер. И действительно, зачем держать в голове то, что компьютер сделает лучше и точно в указанный срок?

А что если необходимо выполнять какие-то простые, но трудоемкие задачи после рабочего дня? Неужели придется оставаться на службе 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
```

При отсутствии даты используется ближайшая возможная. Например, если время больше текущего, то будет установлена текущая дата, если меньше, — 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 12 часов. 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 12:30. 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
```

Подробней об управлении пользователями мы узнаем в *гл. 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, и идентификатор задания. Сообщение будет выглядеть примерно следующим образом:

```
Job 1 at 2005-03-03 12:30
```

С помощью команды `atq` можно увидеть список заданий, поставленных в очередь. Например, результат может быть следующим:

```
1    2005-01-28 12:40 a root
2    2005-01-28 01:00 a root
3    2005-01-30 12:55 a root
```

В первой колонке стоит номер задания, им можно управлять. После этого выводится дата выполнения команды, и в последней колонке — имя пользователя, который создал задачу.

Теперь допустим, что необходимо выполнять в определенное время (после окончания рабочего дня) резервное копирование. А что если в какой-либо день сотрудники задержались на работе, и им необходимо выполнить операции, которые сильно загружают систему. В этом случае резервное копирование будет мешать их работе, и логичнее отложить запуск задания.

Для решения этой проблемы создавайте задачу командой `batch`. Если в момент выполнения задания сервер будет загружен, то работа будет начата, когда нагрузка на сервер будет минимальной, по умолчанию менее 0,8 %.

3.5.2. Планировщик задач

Команда `at` достаточно проста и удобна, но ее задания выполняются однократно, а многие задачи администратора (то же резервное копирование) требуют многократного запуска. Допустим, что вы запланировали резервное копирование ежедневно в 10 часов вечера. Каждый день набирать команду `at` не очень интересно, это надоест через неделю работы и захочется как-то оптимизировать задачу. Создавать файл сценария тоже не слишком удобно, потому что необходимо не забывать его выполнять.

Проблема решается через использование программы cron. Для этого у вас должен быть установлен демон crond, а лучше, если вы включите его в автозагрузку.

Для работы с демоном crond используется программа crontab. Чтобы добавить новую запись в расписание, необходимо выполнить ее без параметров. В ответ на это появится пустая строка, в которой можно задавать шаблон даты и необходимую команду. В общем виде это выглядит как:

```
минуты часы день месяц деньнедели команда
```

День недели указывается числом от 0 до 7. На воскресенье указывают 0 и 7. Это сделано именно так, потому что в различных странах по-разному определяется начало недели — где-то с понедельника, а иногда с воскресенья. В первом случае удобно выставлять значения от 1 до 7, в другом — от 0 до 6.

Если какой-либо параметр не имеет значения, то вместо него необходимо поставить звездочку.

Теперь рассмотрим несколько примеров.

```
00 5 * * * /home/flenov/backup1_script
```

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

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

Эта команда выполняет тот же 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 запустили программу crontab и, не введя команд, нажмете клавиши <Ctrl>+<D>, то все предыдущие задания сотрутся. Будьте внимательны, для выхода из программы без сохранения используйте только клавиши <Ctrl>+<C>.

Помимо этого, у сервиса cron есть несколько дополнительных директорий, упрощающих создание расписаний. Вот распределение выполняемых сценариев 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, если посмотреть на конфигурационный для сервиса cron файл **/etc/crontab**. 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 директории **/etc/cron.monthly** начнут выполняться ежечасно. Так что, названия чисто символические и только по умолчанию. Они легко меняются.

Обратите также внимание, что в этих директориях 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 выходу из этого редактора

изменения вступают в силу мгновенно, а чтобы закончить работу без сохранения изменений, наберите ":q!" и нажмите клавишу <Enter>.

В принципе, все задания cron хранятся в текстовом виде. Для каждого пользователя формируется свой crontab-файл в директории **/var/spool/cron/файл**. Имя созданного файла совпадает с именем пользователя.

Вот пример содержимого crontab-файла:

```
#DO NOT EDIT THIS FILE - edit the master and reinstall.
#(- installed on Thu Jan 27 13:55:49 2005)
#(Cron version--$Id:crontab.c,v2.13 1994/01/17 03:20:37 vixie Exp $)
10 * * * * ls
```

Вы можете редактировать его напрямую, без использования команды `crontab -e`.

3.5.3. Безопасность работ

Напоследок хочется добавить во все преимущества команды `at` ложку дегтя. Злоумышленники очень любят использовать эту инструкцию в своих целях. Например, хакер может завести учетную запись с максимальными правами. Затем настроить команду `at` так, чтобы после выхода она удаляла запись и чистила следы его пребывания в системе.

В каталоге **/etc** есть два файла, которые вы должны настроить:

- at.allow — по умолчанию этого файла может и не быть. Если он существует, то только те пользователи, которые в нем прописаны, могут выполнять команду `at`;
- at.deny — в этом файле перечисляются пользователи, которым явно запрещен доступ к команде `at`.

Подобные файлы есть и для сервиса cron:

- cron.allow — здесь описываются пользователи, которые могут работать с заданиями в cron;
- cron.deny — в этом файле указываются пользователи, которым недоступен сервис cron.

Я не раз говорил и буду повторять, что все настройки должны идти от запрещения. Сначала необходимо все закрыть, а потом позволить только то и лишь тем пользователям, которым посчитаете нужным. Именно поэтому не стоит пытаться внести всех пользователей в список at.deny. Намного корректнее создать файл at.allow и для начала прописать там только свою учетную запись (будет лучше, если это не root-пользователь). Если вслед за этим вы услышите жалобы пользователей о нехватке команды `at`, то сначала убеди-

тесь, что им действительно она нужна, и только после этого прописывайте их учетные записи в файл at.allow.

Ни один лишний пользователь не должен работать с командой `at`. Иногда лучше забыть выполнить команду, чем потерять контроль над системой.

Если вы не запускаете планировщики `at` и `cron`, то я рекомендую убрать сервис crond из автозапуска, а лучше даже удалить. Вы не сможете контролировать то, чем не пользуетесь.

В директории **/etc** есть 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**.

3.6. Настройка сети

После установки ОС Linux легко определяет сетевые карты. В этом у меня еще не было проблем. 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 его правильно настроить.

3.6.1. Адресация в Linux

Основным для Linux является ставший стандартом для Интернета протокол TCP/IP (Transmission Control Protocol/Internet Protocol, протокол управления передачей/протокол Интернета), который уже установлен, и достаточно его только настроить. Если вы еще не встречались с этим протоколом, то советую прочитать соответствующую книгу по этой теме. Мы не сможем здесь затронуть все нюансы TCP/IP, а остановимся только на базовых понятиях.

Для того чтобы сеть работала, нам необходимо, как минимум, настроить следующие параметры:

1. *Указать адрес.* Каждое устройство в сети должно иметь свой адрес. Без этого связь невозможна. Представьте себе, если бы дома не имели своего

адреса, как бы тогда почтальоны доставляли письма? Имена компьютеров для этого не годятся, и об этом мы поговорим в *гл. 11*.

Вся адресация в сети происходит по IP-адресу, который состоит из четырех октетов десятичных чисел (для самой распространенной сейчас 4 версии IP-протокола), разделенных точками. Каждое число не может быть более 255. Если вы подключены к Интернету, то для такого интерфейса может быть установлен адрес, который выдал вам провайдер.

Для подключения по локальной сети адреса задаются самостоятельно. Я рекомендую для примера ограничиться адресами вида 192.168.1.x, где x — это число от 1 до 254 (значения 0 и 255 имеют специальное назначение). Каждому компьютеру должен быть присвоен свой адрес, отличающийся последней цифрой. Третий октет может быть любым, но обязательно одинаковым для всех компьютеров вашей сети. Я у себя использую число 77, т. е. адреса машин имеют вид 192.168.77.x.

2. *Установить маску подсети.* В сочетании с IP-адресом используется номер, называемый маской подсети, позволяющий разбить сеть на более мелкие сегменты (узлы). Из чего состоит ваш домашний адрес? Это город, улица и дом. Сеть имеет только две характеристики — номер сети и номер компьютера внутри нее. Маска определяет, какая часть в IP-адресе относится к сети, а что характеризует компьютер.

   Чтобы понять назначение маски, необходимо каждое число перевести в двоичную систему. Для примера рассмотрим маску 255.255.255.0, она в бинарном виде выглядит так:

   ```
   11111111.11111111.11111111.00000000
   ```

   Теперь переведем в двоичную систему IP-адрес 192.168.001.001:

   ```
   11000000.10101000.00000001.00000001
   ```

   Нужно сопоставить IP-адрес и маску. Там, где стоят единицы — это адрес сети, а нули выделяют адрес компьютера в сети. В маске единицы обязательно должны идти слева, а нули — справа. Нельзя чередовать единицы с нулями. Следующая маска является корректной:

   ```
   11111111.11111111.00000000.00000000
   ```

   А вот такая будет ошибочной:

   ```
   11111111.11111111.00000000.11111111
   ```

   Справа от нулей не может быть единиц.

   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 т. к. под него в данном случае отведено одно число, максимальное значение кото-

рого 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

В данном случае 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. Машины в разных сетях не видят друг друга. Чтобы они смогли взаимодействовать, необходимо специальное устройство (маршрутизатор), которое объединяет разнородные 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 ответа можно увидеть в листинге 3.4.

Листинг 3.4. 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)
```

Как видите, в листинге 3.4 показано два интерфейса: eth0 и lo. Первый из них — это реальная сетевая карта, имя которой в общем виде имеет вид ethX,

где X — число, означающее номер устройства связи в системе. Нумерация начинается с нуля. Если у вас в компьютере две сетевые карты, то их имена будут eth0 и eth1.

Второй интерфейс всегда имеет имя lo (loopback), IP-адрес 127.0.0.1 и маску 255.0.0.0. Этот интерфейс существует в любой системе с сетевой картой и имеет именно этот IP. В принципе, этот адрес ни на что не указывает и не входит ни в какую сеть. Его используют для тестирования и отладки сетевых приложений. Данный интерфейс часто называют петлей, потому что он замыкает на себя. Все пакеты, которые отправляются на этот адрес, посылаются вашему компьютеру.

Помимо сведений о конфигурации сетевых интерфейсов, команда выдает еще много полезной информации, например, количество отправленных и полученных пакетов (параметры `RX` и `TX`).

Есть еще один интересный адрес, который можно увидеть у сетевой карты eth0, — параметр `HWaddr` (Hardware Address, аппаратный адрес). Его еще часто называют MAC-адресом (Media Access Control, управление доступом к среде). Это 48-разрядный серийный номер сетевого адаптера, присваиваемый производителем. Он уникален, потому что у каждого изготовителя свой диапазон адресов. Так как интерфейс lo создан программно (реально не существует), у него не может быть аппаратного адреса.

3.6.3. Изменение параметров сетевого подключения

С помощью ifconfig можно не только просматривать параметры сетевых подключений, но и изменять их. Для этого нужно указать два параметра:

- сетевой интерфейс, параметры которого нужно изменить;
- параметры.

Общий вид команды выглядит так:

```
ifconfig ethX параметры
```

Рассмотрим значения, которые вы можете использовать:

- `down` — остановить интерфейс. Например, для завершения работы сетевой карты eth0 выполните команду: `ifconfig eth0 down`. Если после этого исполнить директиву `ifconfig` без параметров, то в результирующем списке сетевого интерфейса eth0 не будет видно;
- `up` — включить интерфейс, если он был остановлен. Например, если вы хотите восстановить работу сетевой карты eth0, выполните команду: `ifconfig eth0 up`;

- 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`. Одной командой можно 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 директивы `ifconfig`, с которыми вам придется сталкиваться в реальной жизни. Более подробную информацию можно получить, если выполнить команду `man ifconfig`.

3.6.4. Базовые настройки сети

С помощью команды `hostname` можно просмотреть имя компьютера (хоста). Выполните эту команду и перед вами появиться имя, которое вы задали во время установки. Чтобы изменить его, нужно указать новое имя в качестве параметра. Например, следующая команда устанавливает хост "server":

```
hostname server
```

Основные настройки сети находятся в файле **/etc/sysconfig/network**. Давайте посмотрим на его содержимое:

```
cat /etc/sysconfig/network
```

В результате на экране появится примерно следующая информация:

```
NETWORKING=yes
FORWARD_IPv4=true
HOSTNAME=FlenovM
```

Нет смысла изменять какие-либо из этих параметров вручную, когда есть специализированные утилиты. Этот файл я показал вам только для примера.

3.7. Подключение к сети Интернет

К первоначальным настройкам системы я отношу и подключение к Интернету. Если лет 10 назад это было диковинкой и дорогим удовольствием, то сейчас Интернет стал неотъемлемой частью любого компьютера. Трудно себе представить жизнь без общения и обмена информацией.

Всемирная сеть является громадным хранилищем всевозможных данных. Здесь вы сможете найти различное программное обеспечение или документа-

цию. Впоследствии я дам ссылки в Интернете на программы, которые смогут облегчить жизнь, и их необходимо будет скачать *(см. приложение 2)*. Без доступа в сеть это невозможно.

Я не стану описывать все возможные настройки соединений, потому что конкретные нюансы вы сможете узнать у вашего провайдера. Нас будет интересовать соединение через простой модем.

Если вы установили графическую оболочку, то в ней соединение с Интернетом создать легко. Да и работать с Web-страницами в графическом режиме (в браузерах типа Mozilla) намного проще, удобнее и приятнее. Именно для этого вы обязательно должны настроить графическую оболочку, но пользоваться ею необходимо в крайнем случае и только на рабочей станции, но не на действующем сервере.

На данный момент самой простой и удобной программой для создания интернет-соединения является Kppp. Для ее запуска вызовите главное меню Linux и выберите пункт **Интернет/Kppp**. Должно открыться окно, как показано на рис. 3.5.

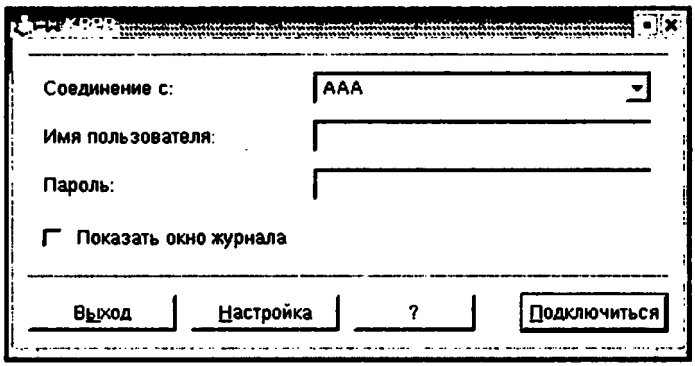

Рис. 3.5. Главное окно программы Kppp

Для начала щелкните кнопку **Настройка** для указания параметров нового соединения. На это действие перед нами открывается диалоговое окно с вариантами настройки: использование мастера или ручной режим. В мастере есть список крупных провайдеров разных стран, и достаточно только выбрать нужную стоку и указать пару дополнительных значений. Если ваш провайдер услуг Интернета отсутствует, то придется делать настройку самостоятельно.

В ручном режиме появляется окно, в котором чаще всего достаточно указать только номер телефона для дозвонки. Остальные параметры в большинстве случаев используют значения по умолчанию.

Настроив соединение, достаточно только ввести имя пользователя и пароль, которые вам дал поставщик услуг, и нажать кнопку **Подключиться**. Если все

указано верно, то через несколько секунд (минут) вы окажетесь во всемирной сети.

3.8. Обновление ядра

Обновление программ позволяет получать новые возможности и исправлять ошибки, сделанные программистами в предыдущих версиях. Основа Linux — это ядро, и оно обновляется очень часто за счет динамичного развития этой ОС. Не пугайтесь ошибок, они есть всегда и везде, и мы еще поговорим об этом *(см. разд. 14.1)*. Вы должны уметь устанавливать новое ядро в свою систему.

Большинство программ в настоящее время реализованы в виде пакетов RPM, которые очень легко инсталлировать. То же самое относится и к ядру ОС. Но на практике самые свежие версии ядра поставляются в исходных кодах. В этом случае процесс установки усложняется, но зато появляется возможность настроить систему на максимальную производительность. Вы можете включить в ядро только то, что необходимо, и оптимизировать под конкретное железо.

Производители дистрибутивов включают в поставку универсальное ядро, которое сможет одинаково хорошо работать на различных платформах. При этом во всех дистрибутивах, которые я видел, включена поддержка модулей. Это очень удобно, но не всегда хорошо.

Еще несколько лет назад взломщики очень часто использовали подмену системных файлов, чтобы встроить в них сплоиты (программа, позволяющая использовать уязвимость) или потайные двери (backdoor). Для борьбы с такой подделкой было разработано множество утилит, которые запрещают изменение системных файлов и следят за их контрольной суммой. В случае каких-либо трансформаций подается сигнал тревоги.

Хакеры, недолго думая, перешли на использование модулей к ОС Linux. Их проследить сложнее, а результат от использования тот же, да и задачи они решают любые. Запретив использование модулей, мы закрываем эту дыру в безопасности, но вы должны учитывать, что могут возникнуть проблемы в работе ОС. Некоторые производители железа или системных утилит любят использовать модули. Это и понятно, ведь их установка проще и позволяет получить необходимые возможности без перекомпиляции ядра. Но мы же знаем, что безопасность и удобство — несовместимые понятия.

3.8.1. Подготовка к компиляции

Прежде чем выполнять какие-то действия по обновлению ядра, нужно подготовиться к самому худшему, а именно — к краху системы. Да, неправильные

действия в самом деле могут нарушить работу или сделать невозможной загрузку системы. Ядро — это основа, и если что-то указать неправильно, то оно может работать некорректно.

Если на вашем сервере уже есть какие-либо данные, то следует сделать их резервную копию. Помимо этого, приготовьте загрузочную дискету, которая может пригодиться в непредвиденной ситуации (например, в случае нарушения загрузочной записи).

Для создания загрузочной дискеты необходимо выполнить следующую команду:

```
/sbin/mkbootdisk ver
```

В данном случае `ver` — это номер версии ядра, установленного в вашей системе. Если вы не знаете текущую версию вашего ядра, выполните команду:

```
uname -r
```

На моем тестовом сервере на данный момент установлено ядро 2.4.18-5asp. Чтобы создать загрузочную дискету для него, выполняем:

```
/sbin/mkbootdisk 2.4.18-5asp
```

Если неверно указать версию, то дискета не будет создана, потому что программа ищет необходимые для загрузочной дискеты файлы в директории **/lib/modules/ver**, где ver — это номер версии. В моем случае это директория **/lib/modules/2.4.18-5asp**.

3.8.2. Обновление ядра из RPM-пакета

Самый простой способ установить новое ядро — использование RPM-пакета. Установка такая же, как и любой другой программы. Для обновления ядра можно выполнить команду:

```
rpm -Uvh ИмяПакета
```

Если вы хотите установить новое ядро, то ключ `U` необходимо заменить на ключ `i`. ОС Linux удобна тем, что можно одновременно установить несколько ядер. Правда, загрузить можно только одно из них.

Из RPM-пакета устанавливаются только все необходимые файлы, модули и загрузчик, но чтобы можно было загрузиться с новым ядром, необходимо еще прописать ядро в загрузчик LILO.

Установка из RPM-пакета дает нам доступ к новым функциям и исправляет старые ошибки. Возможности ядра остаются теми же, что заложил разработчик. Максимальных преимуществ от обновления можно добиться только при компиляции ядра.

3.8.3. Компиляция ядра

При установке из RPM-пакета мы получаем модульное ядро, в котором драйверы устройств могут быть как скомпилированы в одно целое с ядром, так и загружаться отдельно. Такое ядро медленнее в работе, но позволяет обновлять драйверы простой заменой модулей.

При компиляции можно выбрать и монолитное ядро. В этом случае в него будут встроены все необходимые драйверы, что повысит производительность, но это сделает невозможным обновление без перекомпиляции всего ядра.

Я рекомендую вам разобраться с компиляцией, потому что все свежие ядра выходят в исходных кодах, а издание RPM-пакетов иногда запаздывает на неделю и более. Все это время ваша система будет уязвимой.

Как правило, ядро поставляется в виде tar-архива. Сначала его надо разархивировать:

```
tar xzvf linux-2.6.10-rc2.tar.gz
```

Имя архива в вашем случае может отличаться. Я беру самое новое на момент написания книги ядро, которое специально скачивал с сайта **www.redhat.com**.

Архив распаковывается в директорию **linux-2.6.10-rc2** (имя архива без расширения tar.gz). Необходимо перейти в этот каталог, чтобы выполнять дальнейшие действия по компиляции из этой директории.

Для начала нужно сконфигурировать ядро, т. е. указать, что мы хотим получить в результате. Для этого можно использовать одну из четырех утилит:

1. `oldconfig` — сценарий, который устанавливает значения по умолчанию без нашего ведома. Для вызова используйте команду `make oldconfig`.
2. `config` — сценарий, который в командном интерпретаторе задает вам вопросы о параметрах будущего ядра и в зависимости от ваших ответов формируется конфигурационный файл для компиляции. Для вызова используйте команду `make config`.
3. `menuconfig` — текстовая утилита (рис. 3.6). Наиболее удобный вариант конфигурирования из консоли. Для вызова используйте команду `make menuconfig`.
4. `qconf` (`xconfig`) — графическая утилита (рис. 3.7). Наиболее удобный вариант конфигурирования ядра из графической оболочки Linux. Для вызова используйте команду `make xconfig`.

Во время конфигурации с помощью любой из вышеперечисленных утилит вы должны указать, нужна ли поддержка загружаемых модулей.

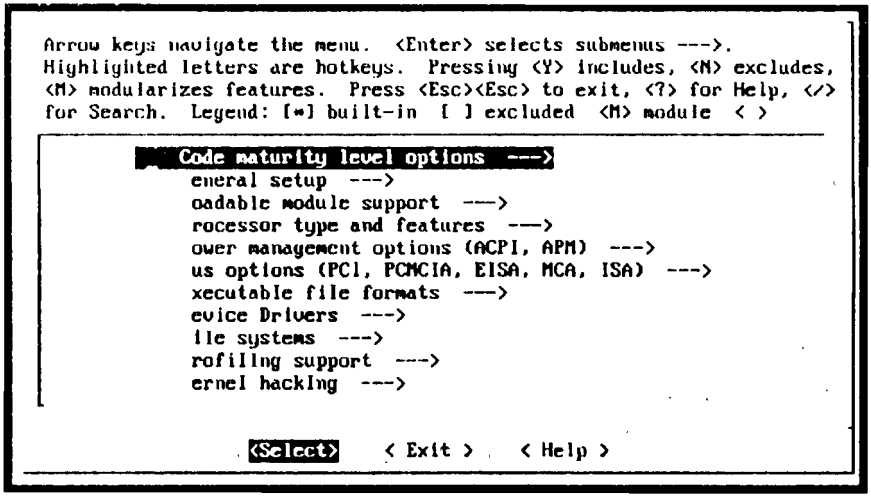

Рис. 3.6. Текстовая утилита конфигурирования ядра menuconfig

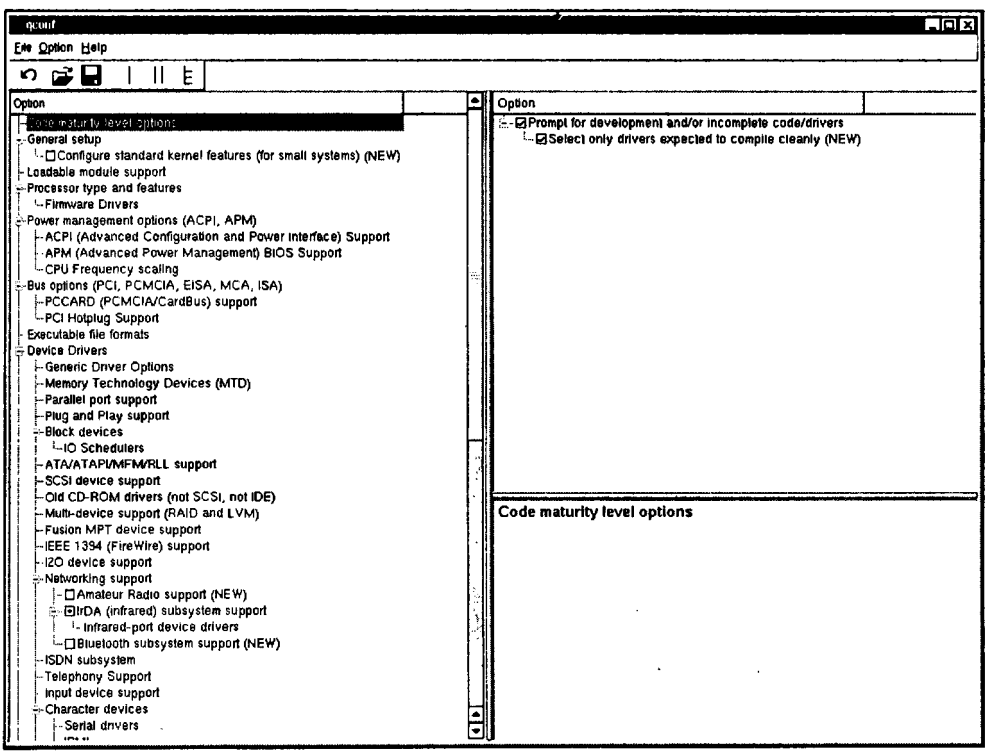

Рис. 3.7. Графическая утилита конфигурирования ядра qconf

Если вы хотите иметь два разных ядра одной и той же версии, например, с поддержкой модулей и без нее, то необходимо открыть файл Makefile и в параметре EXTRAVERSION указать различные значения:

- EXTRAVERSION=-rc2-module — при компиляции ядра с модулями;
- EXTRAVERSION=-rc2-nomodule — при компиляции ядра без модулей.

Лучше всего указывать в этом параметре, чем именно будет отличаться ядро. Например, помимо -rc2-module можно вставить дополнительное пояснение, но оно должно быть максимально коротким.

Теперь выполняем команды подготовки к компиляции:

```
make dep
make clean
```

Следующая команда отнимет достаточно много времени, потому что она будет компилировать непосредственно ядро. Можно отправляться готовить кофе и пить его медленно и печально. Если у вас слабый процессор и памяти менее 256 Мбайт, то процедура будет долгой.

Итак, для компиляции ядра выполним команду:

```
make bzImage
```

Во время компиляции вы увидите список собираемых в ядро модулей. Их очень много, поэтому процесс продолжительный.

Если вы выбрали компиляцию ядра с использованием загружаемых модулей, то следующие две команды должны выполнить эту операцию:

```
make modules
make modules_install
```

Если вы запретили использование модулей, то эти команды можно пропустить, потому что от них толку мало, а выполняются они долго.

Все модули располагаются в директории **/lib/modules/**. С помощью первой команды мы скомпилировали модули нового ядра, а вторая скопирует их в каталог **/lib/modules/**. Здесь вы найдете директории модулей для каждой из версий ядер, установленных в системе.

Теперь инсталлируем скомпилированное ядро. Для этого выполняем следующую команду:

```
make install
```

Эта команда скопирует все необходимые файлы для загрузки на свои места. Загляните в директорию **/boot**. Здесь можно найти несколько файлов, в том числе и загрузчик для новой версии ядра. Их легко можно определить по но-

меру. Например, я компилировал ядро 2.6.10, и у меня появились файлы vmlinuz-2.6.10 и initrd-2.6.10.img.

3.8.4. Настройка загрузчика

Теперь посмотрим, как можно настроить загрузку нового ядра, оставив при этом возможность пользоваться старыми версиями. Для этого нужно отредактировать файл **/etc/lilo.conf**, добавив в конец файла следующие строки:

```
image=/boot/vmlinuz-2.6.10
label=Linux Kernel 2.6.10
initrd=initrd-2.6.10.img
read-only
root=/dev/hda0
```

В первой строке находится путь к ядру. Вы должны указать правильный путь, а точнее, имя файла, которое будет отличаться номером версии. Параметром `label` задается заголовок нового ядра, который вы будете видеть при загрузке системы. Параметр `initrd` определяет загрузчик. Последняя строка указывает корневой диск. Введите такое же значение, как и у старого ядра. В вашем загрузчике уже должна быть подобная строка, и она одинакова для обеих систем.

После изменения конфигурационного файла **/etc/lilo.conf** необходимо прописать изменения в загрузочную область. Для этого выполните команду `lilo`.

Перезагрузите систему, и теперь при старте компьютера вы сможете выбирать, какое ядро загружать. При этом будут использоваться одни и те же конфигурационные файлы, что очень удобно. Если новое ядро окажется неработоспособным, достаточно перезагрузить компьютер со старым ядром и продолжить работу до выхода обновлений.

3.8.5. Работа с модулями

Преимущество модульной сборки ядра заключается в том, что вы можете изначально включить только самые необходимые функции и тем самым уменьшить потенциальные уязвимости, которыми может воспользоваться хакер. Но при этом мы должны иметь возможность управлять модулями (так же, как и сервисами).

Система должна загружаться только с теми модулями, которые применяются. Все остальные должны подгружаться по мере необходимости и отключаться, когда не используются.

lsmod

С помощью команды `lsmod` можно увидеть список загруженных модулей. Результат выполнения инструкции имеет примерно следующий вид:

```
Module          Size Used by  Not tainted
binfmt_misc        7428  1
autofs            11812  0 (autoclean) (unused)
tulip             42240  1
ipchains           42216  6
ide-cd            30240  0 (autoclean)
cdrom             32000  0 (autoclean) [ide-cd]
ext3              62284  1
jbd              39804  1 [ext3]
```

modinfo

Очень трудно разобраться, какие модули нужны в системе, а какие нет. А ведь необходимо загружать только то, что действительно используется, иначе увеличивается время старта системы, понапрасну расходуются ресурсы компьютера и т. д. Так как же в этом разобраться? Необходимо знать каждый модуль в лицо и понимать, для чего он нужен.

Получить информацию о модуле помогает команда `modinfo`. В качестве параметра нужно передать имя интересующего модуля, и на экране будет выведена информация о нем. Например, следующая команда запрашивает у системы информацию о модуле ext3:

```
modinfo ext3
```

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 расположения файла, описание, автор, лицензия и т. д. Количество информации сильно зависит от модуля, и если честно, в некоторых случаях она настолько скудная, 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 нужно 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. Иначе как раз он и может стать причиной взлома.

ГЛАВА 4

Управление доступом

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

Обычному пользователю вы должны выделять ограниченные права, которые позволят выполнять только необходимые операции. Кроме того, вы должны минимизировать количество пользователей с большими привилегиями, потому что такие учетные записи требуют особо пристального внимания и наблюдения. Если под привилегированной учетной записью вошли в систему с компьютера, где владелец записи не мог находиться, то это укажет на потенциальную опасность или взлом.

После увольнения сотрудника вы должны удалить его учетную запись, чтобы недовольный этим фактом он не уничтожил важные данные.

Для работы с командами управления доступом вы должны обладать правами администратора. Для этого можно войти в систему как пользователь root или использовать директиву `su`. В *разд. 4.16* мы рассмотрим еще один способ — использование утилиты 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 быть одно из следующих значений:

- знак тире (—) — обычный файл;
- буква "d" — каталог;
- буква "l" — символьная ссылка;
- буква "s" — сокет;
- буква "p" — файл FIFO (First in first out, первый вошел — первый вышел).

После этого в каждой строке идет три группы символов rwx, определяющих 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). Установленная буква говорит о разрешении соответствующего действия. Рассмотрим несколько вариантов.

Первая строка в нашем примере `drwx------`. Первый символ d, а значит, это директория. Потом идут три символа `rwx`, т. е. хозяин файла может читать, записывать и исполнять директорию. Вместо остальных шести символов стоят знаки тире, значит, у пользователей группы `FlenovG` и у всех остальных нет прав.

Вторая строка — `drwxr-xr-x`. Это снова директория. Потом стоит комбинация `rwx`, а значит, владельцу разрешены все операции. Следующая тройка, соответствующая группе, равна `r-x`, а стало быть, возможно чтение и исполнение. В соответствии с последней тройкой всем остальным пользователем также доступно только чтение и исполнение.

Последняя строка в примере содержит права доступа `-rwxr-xr--` для файла (первый символ — это знак тире). Хозяин файла имеет полный доступ к нему (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 воспринимать как последовательность нулей и единиц. Если в определенном месте стоит 1 (указан один из символов r, w или x), то операция разрешена. Если 0 (находится знак тире), то действие запрещено. Давайте попробуем записать права `rwxr-xr--` в виде нулей и единиц. Установите вместо букв единицы, а вместо тире — нули. Должно получиться 111101100. Ра-

зобьем эту комбинацию на три части 111, 101 и 100. Теперь каждую тройку переведем в восьмеричную систему по следующей формуле:

```
Цифра1 * 4 + Цифра2 * 2 + Цифра3
```

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

- 0 — запрещено все;
- 1 — разрешено выполнение;
- 2 — разрешена запись;
- 3 — разрешена запись и выполнение;
- 4 — разрешено чтение;
- 5 — разрешено чтение и выполнение;
- 6 — разрешено чтение и запись;
- 7 — разрешено все.

Попробуйте теперь с помощью этого списка определить возможности, которые предоставляет число 754. Каждый разряд нужно рассматривать в отдельности. Сравните полученный результат с символьным представлением `rwxr-xr`. Должно выйти одно и то же.

Внимание!

Для того чтобы иметь право создания или удаления файлов, необходимо иметь разрешение записи на директорию. Это немного сбивает с толку начинающих администраторов, т. к. им непонятно, почему при наличии всех прав на файл его нельзя удалять.

4.1.1. Назначение прав

Для изменения режима доступа на объекты файловой системы используется команда `chmod`. В ней можно указывать новые права на объект как в символьном (применяется для изменения относительно текущего состояния), так и в числовом виде (абсолютное задание). Для начала рассмотрим символьный режим:

```
chmod параметры права файл
```

Параметры могут включать комбинацию значений изменения прав по категориям пользователей:

- u — владельца;
- g — группы;
- o — остальных пользователей;
- a — все права (то же самое, что передать значение ugo).

Перед указанием прав можно задать режим их изменения относительно существующих:

- + — 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 — setuid- или setgid-бит;
- t — sticky-бит. В этом случае только владелец файла и каталога сможет выполнить удаление;
- u — всем пользователям, как и у владельца;
- g — всем пользователям, как и у группы;
- o — для остальных, как у пользователей, не входящих в группу файла и не являющихся его владельцем.

В случае с числовым представлением команда выглядит следующим образом:

```
chmod права файл
```

Права передаются в виде восьмеричного числа из четырех разрядов:

- первый — определяет дополнительный бит и может принимать одно из значений:
  - 1 — бит принадлежности;
  - 2 — setgid-бит;
  - 4 — setuid-бит;
- второй — права пользователя. Это число может быть от 0 до 7;
- третий — права группы. Значение в диапазоне от 0 до 7;
- четвертый — права остальных пользователей. Это число может быть от 0 до 7.

Например, мы хотим, чтобы владелец и группа имели все права (число 7), а остальные пользователи могли только выполнять файл (число 1). Значит, команда будет выглядеть следующим образом:

```
chmod 771 filename
```

Число 771 в символьном виде соответствует правам `rwxrwx--x`. Следующая команда отменит возможность чтения файла группой:

```
chmod g-r text
```

После этой команды права доступа на файл станут `rwx-wx--x`. Теперь давайте запретим всем запуск файла. Для этого можно выполнить команду:

```
chmod ugo-x text
```

или

```
chmod a-x text
```

После наших манипуляций права доступа на файл станут `rw--w----`.

4.1.2. Владелец файла

Для изменения владельца файла существует команда `chown`:

```
chown имя файл
```

Через параметр `имя` определяется пользователь, которому нужно передать права на указанный файл. Например, давайте сделаем владельцем файла test администратора root. Для этого нужно выполнить следующую команду:

```
chown root test
```

Изменить можно и группу, к которой принадлежит файл. Для этого выполните команду `chgrp`:

```
chgrp имя файл
```

Здесь задается имя группы, которой предоставляются права на указанный файл. Например, для файла test укажите группу администратора root с помощью такой команды:

```
chgrp root test
```

4.1.3. Правила безопасности

При назначении прав на доступ к файлам и папкам вы должны следовать принципу минимализма, описанному в *разд. 2.10.1*. Чтобы это правило действовало, по умолчанию должно быть запрещено все. Открываем доступ только на то, что необходимо, и ничего лишнего. Если файл не должен быть

виден пользователю, то нельзя разрешать даже его отображение в дереве каталогов.

Любые лишние объекты могут оказаться фатальными для безопасности системы не только с точки зрения взлома, но и утечки информации. Например, файлы бухгалтерской отчетности должны быть доступны только для тех, кто с ними работает. Если показать эти документы всем, то финансовые данные, возможно, станут достоянием общественности, и это нежелательным образом отразится на благосостоянии компании.

Самое главное для защиты вашей системы — не дать пользователям возможность изменять системные файлы. В Linux основные конфигурационные файлы находятся в каталоге /etc. Только администратор root должен иметь право их модифицировать. Производители дистрибутивов настраивают систему по умолчанию именно так, и не следует повышать статус пользователей без особой надобности.

4.1.4. Права по умолчанию

Когда пользователь создает новый файл или директорию, то им назначаются права по умолчанию. Давайте разберем 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 группы и все остальные — только просматривать. В старых системах и некоторых дистрибутивах права могут быть равны `-rw-rw-r--`, а значит, пользователи группы тоже смогут корректировать файл. Такие права нарушают главное правило безопасности. Но в любом случае все получают возможность читать файл. Это разрешено.

Такая политика неверна, потому что если вы создадите файл, который хранит конфиденциальные данные, то информация будет доступна для всеобщего обозрения. И если вы забудете понизить права, то любой сможет увидеть и прочитать файл.

Ситуацию можно изменить, если понимать, как создаются права для нового файла. Они рассчитываются на основе маски, текущее значение которой определяется командой `umask`. В результате 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 директорий — 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--`, а при значении `0022` формула изменится на 666-0022=644, что будет означать `-rw-r--r--`.

Для директорий 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 777-002=755, а это соответствует правам `drwxr-xr-x`, т. е. все пользователи смогут просматривать директории и увидят содержащиеся в ней файлы.

Все это не есть хорошо. Если владелец должен иметь доступ, достаточный для полноценной работы с файлами и директориями, то остальные вообще не должны иметь прав. Эту ситуацию можно исправить изменением маски. Я рекомендую установить ее в `077`. В этом случае для директорий права будут определены как 777-077=700 (или `drwx------`), а для файлов — 666-077=600 (или `-rw-------`). Тогда доступ к файлу имеет только владелец. Все остальные — отдыхают.

При расчете прав на файл можно подумать, что я ошибся, ведь 666-077 далеко не равно 600. Почему же так получилось? Просто вычитание происходит поразрядно, т. е. первая цифра 6 в числе минус первая цифра 0 в маске, затем операция производится со вторыми цифрами (6-7) и т. д. Если какой-либо результат получается отрицательным, то он округляется до нуля.

Вот такое положение дел нас уже устраивает. Чтобы установить новую маску, выполните команду `umask маска`. В нашем случае это будет `umask 077`.

4.1.5. Права доступа к ссылкам

В *разд. 3.1.3* мы рассматривали жесткие и символьные ссылки на файлы. Для начала вспомним права на жесткие ссылки:

```
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
```

Как видите, права абсолютно идентичны. Я надеюсь, что вы другого и не ожидали, ведь у ссылок дескрипторы одинаковые.

С символьными ссылками дело обстоит куда хуже. Вот пример основного файла и символьной ссылки на него из того же *разд. 3.1.3*:

```
913021 -rw-r--r-- 1 root root 519 Feb 22 12:19 link.txt
913193 lrwxrwxrwx 1 root root 8 Feb 22 12:40 symbol.txt -> link.txt
```

Первая строка содержит информацию о файле, а во второй — находится символьная ссылка. Как видите, у нее открыт абсолютно полный доступ. Если

создать ссылку для файла **/etc/shadow** и не изменить ее права, то можете попрощаться с паролями, их украдут или обнулят. Помните, что любая операция по изменению файла символьной ссылки затрагивает непосредственно сам файл.

Если вы решили использовать символьные ссылки, то не забудьте об особенности формирования прав доступа на файлы. Можете даже выбить на мониторе надпись: "Ссылки на файлы создаются с полными правами!!!"

4.2. Управление группами

Начнем изучение вопроса с создания групп. Что это такое? 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 пользователя? А может быть писать собственную программу? Оба способа неудобны и требуют больших усилий.

Что если объединить этих пользователей в группу, а уже ей дать право на использование определенного файла. Впоследствии, если нужно запретить доступ, то одной командой отключаем разрешение для группы, и все 500 пользователей больше не смогут работать с файлом. Удобно? Даже очень.

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 используется команда `groupadd`. Она выглядит следующим образом:

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

После имени команды можно указывать следующие параметры:

- `-g gid` — идентификатор группы. Это неотрицательное число, которое должно быть уникально. Если вы ввели значение, которое уже используется в системе, то для нормальной отработки команды нужно добавить еще и ключ `-o`. В большинстве случаев идентификатор вообще не нужно указывать. Тогда система возьмет первое свободное значение, начиная с 500;
- `-r` — этот ключ указывает на необходимость создания системной группы. Идентификаторы таких групп находятся в диапазоне от 0 до 499. Если

в явном виде не указано значение параметра `-g` , то будет выбрано первое свободное число менее 500;

- `-f` — блокирует создание групп с одинаковыми именами. Если указать этот ключ, то команда отработает, но новая группа не сформируется, а уже существующая не будет обновляться.

Если какие-либо параметры не указаны, то будут использоваться значения по умолчанию. Рассмотрим примеры создания групп (после знака "#" идет комментарий, поясняющий работу команды):

```
groupadd testgroup1 # Создать группу testgroup1 с ID по умолчанию
groupadd -g 506 testgroup2 # Создать группу testgroup1 с ID 506
groupadd -r testgroup3 # Создать группу testgroup1 с системным ID
          #(менее 500) по умолчанию
```

Вся информация о группах добавляется в файл **/etc/group**. Откройте его содержимое, например, в MC или наберите в командной строке:

```
cat /etc/group
```

Эта команда выводит на экран содержимое файла, в самом конце которого вы увидите три строки с информацией о добавленных нами группах:

```
testgroup1:x:500:
testgroup2:x:506:
testgroup3:x:11:
```

Файл состоит из 4 колонок: имя группы, пароль, идентификатор, список пользователей. Колонки разделены знаком двоеточия.

В первой группе мы не указывали идентификатор, поэтому система выбрала значение по умолчанию. Во второй — в явном виде задан номер группы. В последней строке идентификатор равен 11, потому что был запрошен номер по умолчанию из системного диапазона (использовался ключ `-r`).

Последняя колонка (после третьего двоеточия) ничего не содержит. Здесь должен быть список пользователей группы, но он пуст, потому что мы еще его не формировали.

4.2.2. Редактирование группы

Для редактирования параметров группы можно напрямую изменять файл **/etc/group**, но я рекомендую лучше 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

Теперь рассмотрим, как можно удалить группу. Для этого используется 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 root-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 используется команда `useradd`. С ее помощью также можно изменить значения по умолчанию, которые будут присваиваться учетной записи.

Команда выглядит следующим образом:

```
useradd параметры имя
```

Параметров очень много, большинство из них вам знакомо по файлу **\etc\password**, который мы рассматривали в *гл. 3*. Рассмотрим каждый аргумент:

- `-c` — простое текстовое описание, которое может быть любым;
- `-d` — домашний каталог пользователя;
- `-e` — дата отключения учетной записи, после которой пользователь станет неактивным, вводится в формате ГГГГ-ММ-ДД;
- `-f` — количество дней до отключения. Похоже на параметр `-e`, только указывается период относительно текущей даты. Если запись нужно отключить сразу после создания, то используйте значение `0`. По умолчанию устанавливается `-1`, что соответствует бесконечности, т. е. запись будет активной всегда;
- `-g` — основная группа, которой будет принадлежать пользователь. Можно указывать как имя, так и идентификатор. В Red Hat каждый пользователь принадлежит какой-либо группе;
- `-G, [...]` — дополнительные группы, в которых будет существовать пользователь. Имена групп перечисляются через запятую;

- `-m` — ключ для создания домашнего каталога пользователя. В такую директорию будут скопированы все файлы из **/etc/skel**;
- `-M` — не создавать домашний каталог. По умолчанию директория формируется в `/home/ИмяПользователя`, чтобы этого избежать, нужно явно прописать в команде запрет;
- `-r` — если указать этот параметр, то в качестве идентификатора будет выбрано число из системной области;
- `-p` — зашифрованный пароль, который можно получить с помощью команды `crypt`;
- `-s` — командный интерпретатор, который будет обрабатывать директивы пользователя;
- `-u` — идентификатор, который должен быть уникальным. Если его не устанавливать, то система выберет свободное значение.

Самый последний параметр — имя создаваемой учетной записи. Давайте рассмотрим, как можно добавить нового пользователя по имени robert, для которого все значения будут установлены по умолчанию:

```
useradd robert
cat /etc/passwd
```

В первой строке мы создаем нового пользователя по имени robert. Вторая строка выводит на экран содержимое файла **/etc/passwd**, где хранится информация о всех учетных записях. И заключительная строка в нем будет выглядеть следующим образом:

```
robert:x:501:501::/home/robert:/bin/bash
```

Вспомните формат этого файла, который мы рассматривали в *разд. 3.3*. Первый параметр — это имя. Затем стоит пароль, который спрятан в теневом файле, поэтому здесь `x`. Далее следуют идентификаторы пользователя и группы. Так получилось, что в обоих случаях свободными оказались номера, равные 501, поэтому идентификаторы одинаковы, но это далеко не всегда так. Потом идет домашний каталог пользователя. По умолчанию все директории создаются в папке **/home** и соответствуют имени пользователя.

Давайте посмотрим файл **/etc/shadow**. Обратите внимание, что в строках пользователей robert и Denver стоит два восклицательных знака, мы не указывали пароль и войти в систему не можем. Я и не советую его задавать при создании пользователя. Это лишние мучения, потому что нужно шифровать его функцией `crypt`, при этом нет гарантии сложности пароля. Лучше изменить его после создания пользователя с помощью команды `passwd`:

```
passwd robert
```

В ответ на это вы увидите приглашение ввести пароль и пояснения о необходимости делать его сложным. Сообщение, которое выдает программа, выглядит следующим образом:

```
Changing password for user robert.

You can now choose the new password or passphrase.

A valid password should be a mix of upper and lower case letters,
digits and other characters. You can use an 8 character long
password with characters from at least 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 следующим образом:

```
Изменяется пароль для пользователя robert.

Сейчас вы можете выбрать новый пароль или идентификационную фразу.

Хороший пароль должен состоять из комбинации заглавных и прописных букв,
цифр и других знаков. Вы можете ввести пароль длиной в 8 символов
с использованием значений 3 и 4 типов или пароль из 7 символов,
сочетающий знаки всех классов. Символы из часто используемых шаблонов
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 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. Злоумышленник может 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 зато намного меньше, чем для шифра типа OLhslu_9&Z435drf. Для нахождения этого пароля словарь не поможет, а полный перебор всех возможных вариантов отнимет годы.

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

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

Ключ `-a` заставляет отображать все файлы (в том числе и системные), а `-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 .gtkrc
drwxr-xr-x    4 robert   robert     4096  Nov 26 16:10 .kde
```

Обратите внимание, что в директории 6 файлов и один подкаталог. Самое интересное находится в третьей и четвертой колонках, где располагаются имя и группа владельца файла соответственно. В обоих столбцах почти везде указано имя robert. Если пользователя с таким именем мы только что создали, то группу — нет. Вспомните *разд. 4.2*. При отсутствии параметров настройки автоматически формируется группа с таким же именем, что и учетная запись, и туда сразу же помещается все необходимое о новом пользователе.

Еще один нюанс. Папка с именем из двух точек (..), указывающая на родительский каталог, принадлежит root. Почему? Пользователь robert — владелец текущей директории **/home/rober**, (он здесь хозяин), но каталог выше **/home** вне его прав.

Все файлы и папки, которые принадлежат учетной записи robert, доступны для чтения и записи. Пользователи группы robert и все остальные могут только просматривать информацию, а разрешения на изменение у них нет.

4.3.1. Файлы и папки нового пользователя

Откуда берутся файлы в папке нового пользователя? При формировании учетной записи в соответствующую домашнюю папку копируются все файлы и подкаталоги из **/etc/skel**. Давайте создадим свой файл в этой директории и посмотрим, попадет ли он в папку нового пользователя? Чтобы ничего не выдумывать, выполним следующую директиву:

```
ls >> /etx/skel/text
```

Здесь задается команда `ls` для просмотра содержимого текущего каталога. Потом идет два символа ">>" и имя файла "text" в папке **/etc/skel**. Такая

запись означает, что результат выполнения команды должен быть помещен в указанный файл. Если файл не существует, то он будет создан. Таким образом, мы подготовили в нужной директории новый файл, содержимое которого нас не особо волнует.

Теперь добавляем нового пользователя и просматриваем содержимое его папки:

```
useradd Denver
ls -al /home/Denver
```

В результате вы увидите, что созданный нами в каталоге **/etc/skel** файл был скопирован в папку нового пользователя. В директориях уже существующих пользователей этого файла не будет.

Эту полезную особенность я использую достаточно часто, чтобы сразу наделить нового пользователя правилами нахождения в системе, необходимыми ему файлами, документацией и т. д.

Среди файлов, копируемых в каталог нового пользователя, есть bash_profile. Это профиль командного интерпретатора **/bin/bash**. В нем можно настраивать некоторые параметры, в том числе и маску. В *разд. 4.1* мы говорили о правах, которые назначаются всем новым файлам пользователя (далеки от идеала), и научились понижать их с помощью команды `umask`.

Зайдите под учетной записью robert и посмотрите маску с помощью команды `umask`. Обратите внимание, что она равна 0002. То есть мы в *разд. 4.1* изменили маску своего пользователя, а robert 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 директориям 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 директорию, то он легко сможет определить имя пользователя, которому она принадлежит, и наоборот.

При создании пользовательских директорий 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
```

Этот файл можно редактировать вручную или воспользоваться командой `useradd` для изменения значений по умолчанию. Как это делать, мы рассмотрим чуть позже.

Единственный комментарий, который я хочу сделать сейчас, — это параметр `GROUP`. Он равен `100`, и по идее все новые пользователи должны попадать в эту группу. Но как мы видели в *разд. 4.3*, это не так. В Red Hat этот параметр игнорируется, и по умолчанию создается новая группа, именно это мы и наблюдали. В других дистрибутивах этот параметр может работать, поэтому проверьте эту возможность, чтобы не оказаться в неудобном положении.

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

Но созданием пользователя команда `useradd` не ограничивается. Дело в том, что если не указать какой-либо параметр, то ему будет присвоено значение по умолчанию. Чтобы внести изменения, нужно сразу после имени команды указать ключ `-D`. После этого могут идти следующие опции:

- `-g` — изменить группу;
- `-b` — установить домашний каталог;
- `-f` — время до отключения;
- `-e` — дата отключения;
- `-s` — оболочка (интерпретатор команд).

Я советую вам не игнорировать возможность указания времени действия учетной записи. Допустим, что к вам пришли с инспекцией и просят дать доступ к базе данных или определенным файлам. Создайте для проверяющих нового пользователя и установите время жизни в 1 день (или более, если инспекция приехала надолго). В этом случае вам не надо напрягать память

или фиксировать в блокнот, что в определенный день требуется удалить такую учетную запись, потому что она сама станет неактивной.

Некоторые администраторы плодят временных пользователей, забывая их удалить. А ведь это достаточно большая дыра в безопасности, т. к. эти учетные записи чаще всего имеют простой пароль. Действительно, зачем запоминать (на один-два дня) что-то типа opihvgdjwbe. Отключив запись (автоматически или вручную), вы закрываете один из проходов в вашу систему и сокращаете количество лазеек. Когда вы возвращаетесь домой, то обязательно закрываете за собой дверь для собственной безопасности. В случае с ОС надо поступать таким же образом, и, проводив временного гостя, нужно обязательно запереть дверь или замуровать ее (удалить из системы).

4.3.3. Редактирование пользователя

Для редактирования параметров учетной записи можно напрямую корректировать файл **/etc/passwd**, но я советую лучше использовать команду `usermod`. У нее такие же ключи, что и у `useradd`, но она не создает пользователя, а изменяет параметры уже существующего.

С помощью `usermod` вы можете добавлять уже имеющегося пользователя в ранее созданную группу. Давайте проделаем такую процедуру с учетной записью robert и определим ее в группу `root`. Это позволит пользователю robert выполнять некоторые административные функции:

```
usermod -G root robert
```

Здесь мы выполняем команду с ключом `-G`, который позволяет указать, членом каких групп должен быть пользователь (в данном случае root). Можно перечислить несколько групп, разделяя их запятыми. Для получения более подробной информации о команде `usermod` выполните `man usermod`.

Параметры у команды `usermod` такие же, как и у `useradd`.

4.3.4. Удаление пользователя

Для удаления пользователя применяется команда `userdel`. В качестве параметра передается только имя учетной записи, которую надо удалить, и можно распрощаться с ней навсегда. Например:

```
userdel Danver
```

Будет получено сообщение об ошибке, если пользователь в этот момент находится в системе.

Необходимо учитывать, что удаление пользователя не уничтожает его домашний каталог, вы должны сделать это вручную. Если указать в команде

ключ -r, то вместе с пользователем будут удалены и его файлы в домашней директории:

```
userdel -r Danver
```

Никогда не указывайте этот ключ. Выполняйте операцию только вручную, просмотрев содержимое каталога и убедившись, что там нет нужных файлов.

К тому же, если для пользователя была автоматически создана группа и в ней никого больше нет, то можно произвести и ее удаление, но сделать это придется командой `groupdel`.

4.3.5. Несколько замечаний

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

#
# If defined, this command is run when removing a user.
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).
#
#USERDEL_CMD       /usr/sbin/userdel_local

#
# If useradd should create home directories for users by default
# On RH systems, we do. This option is ORed with the -m flag on
# useradd command line.
#
CREATE_HOME       yes
```

Здесь находятся интересные установки, которые можно использовать для повышения безопасности. Рассмотрим основные параметры файла:

- MAIL_DIR — директория, в которой будет храниться почта пользователей;
- PASS_MAX_DAYS — максимальный срок жизни пароля;
- PASS_MIN_DAYS — минимальный срок жизни пароля;
- PASS_MIN_LEN — минимальная длина пароля (используется только в команде passwd и игнорируется в useradd). В большинстве дистрибутивов здесь будет стоять 5. Я рекомендую поменять его на число 8. В этом случае нельзя будет установить любимый большинством пользователей пароль типа qwerty;
- PASS_WARN_AGE — срок (в днях) до окончания действия пароля, когда об этом нужно предупредить пользователя;
- UID_MIN — минимальный идентификатор пользовательских учетных записей;
- UID_MAX — максимальный идентификатор пользовательских учетных записей;
- GID_MIN — минимальный идентификатор пользовательских групп;
- GID_MAX — максимальный идентификатор пользовательских групп;
- CREATE_HOME — признак создания пользовательской директории (значение параметра YES, стоит по умолчанию).

4.3.6. Взлом паролей

Я еще раз хочу напомнить о недопустимости использования простых паролей не только для администратора, но и для всех пользователей системы. Если проследить за уязвимостями, которые находят в Linux-системах, то очень часто можно увидеть сплоиты, которые позволяют повысить права хакера от простого пользователя до root. Если взломщик не сможет получить доступ даже в качестве простого пользователя, то и воспользоваться сплоитом будет невозможно.

Сложные пароли должны быть абсолютно у всех пользователей. Если хакер получит доступ к файлу **/etc/shadow** с 1000 записями, то подбор паролей весьма упрощается. Вспомните, как хранятся пароли. Они зашифрованы необратимым образом. Это значит, что при простом переборе каждый возможный вариант тоже шифруется, а потом сравнивается с результатом из файла **/etc/shadow**. За счет того, что кодирование отнимает достаточно много процессорного времени, сопоставление становится слишком продолжительным.

Подбирать сложно, если вы сравниваете только с одной записью. А если в системе 1000 пользователей, то достаточно один раз зашифровать возможный вариант пароля и потом соотнести его с 1000 записями в файле паролей. Вероятность попадания увеличивается в несколько раз и подбор упрощается.

Когда хакеры получают файл **/etc/shadow**, то первым делом запускается проверка всех записей, в которых имя пользователя и пароль одинаковы. Вы не поверите, но такое встречается очень часто, и если файл паролей большой, то с вероятностью 0,9 можно сказать, что хакер найдет такую запись.

Если это не помогло, то в ход идет перебор всех часто используемых слов. Вот тут уже вероятность попадания близка к 100 %, потому что из десяти пользователей один обязательно будет новичком и установит простой пароль. Вы должны проводить обучение каждого нового пользователя (в частности, по вопросу указания пароля) и самостоятельно запускать программу сопоставления паролей с часто используемыми словами. Если вам удалось подобрать, то хакер тем более сделает это.

4.4. Типичные ошибки распределения прав

Строгое распределение прав может значительно обезопасить систему. При правильной регламентации доступа большинство взломов могут оказаться неэффективными. Например, однажды в Интернете появилось сообщение, что один из сервисов ОС Linux содержит ошибку. Благодаря хорошему распределению прав мой сервер оказался защищенным от проникновения через эту дыру. Злоумышленник мог пробраться на сервер, но ничего изменить или

удалить нельзя, потому что внешним пользователям этого сервиса все файлы открыты только для чтения.

Итак, если у вас хорошо настроены права доступа, то это может оказаться непреодолимой преградой для злоумышленника.

Давайте рассмотрим классический пример с файлами и каталогами. Допустим, что у вас на каталог поставлены максимальные права `drwxrwxrwx` (или 777), а на все файлы в нем — `-rw-------`. По идее, только владелец файла может его модифицировать, но это не совсем так. Да, злоумышленник не сможет изменить сам файл, но список документов в директории ему доступен (разрешены чтение и корректировка). Благодаря этому взломщик просто удалит нужный файл и создаст другой с новыми правами без каких-либо преград.

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

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

Допустим, что 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` для всех пользователей стоит "`t`". Как раз он и указывает на установленный sticky-бит. Теперь попробуйте удалить из этой директории файл, принадлежащий другому пользователю. Вы увидите сообщение "rm: cannot unlink ‘имя файла’: Operation not permitted".

Используйте этот бит на всех открытых папках. Некоторые хакеры, добравшись до диска и не обретя доступа к закрытой информации, начинают уничтожать все, что видят. С помощью sticky-бита взломщик сможет удалить только то, что создал сам, и ничего лишнего.

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

Проверьте, если в вашей системе это не так, то установите его самостоятельно, чтобы никто не смог удалить чужие временные файлы.

4.5. Привилегированные программы

В *гл. 3* я уже намекал про существование еще двух битов доступа — SUID и SGID, и теперь пора с ними познакомиться поближе. Допустим, что пользователя необходимо ограничить в правах, чтобы он не натворил бед, но при этом дать возможность запускать программу, к которой требуется специальный доступ. В этом случае можно установить SUID-бит, тогда и программа сможет работать (от имени владельца), и у пользователя не будет лишних привилегий.

SUID-бит можно установить командой `chmod` с параметром `u+s`:

```
chmod u+s progname
```

Если теперь просмотреть права доступа к файлу, то они превратятся в `-rwsr-xr-x`. Как видите, появилась буква "s" на том месте, где должно быть разрешение на запуск (символ "x") для владельца файла.

Бит SGID похож на SUID, но он позволяет запускать программу с правами группы владельца файла. Этот бит устанавливается подобным образом командой `chmod` с параметром `g+s`:

```
chmod g+s progname
```

В этом случае права доступа к файлу будут `-rwxr-sr-x`. Вместо символа "x" (право на запуск для группы владельца файла) появилась буква "s".

Привилегии SUID и GUID достаточно удобны и полезны, но, с другой стороны, они таят в себе очень много проблем. Например, гость, обладающий минимальными правами, запускает программу с установленным битом SUID, владельцем которой является пользователь root. Это значит, что программа будет работать с правами root, а не гостевыми параметрами пользователя. Если в ней окажется ошибка, через которую можно выполнять команды на сервере, то эти директивы будут реализовываться от имени владельца программы, т. е. root. Таким образом, даже если взломщик сам не имеет возможности претворять в жизнь команды, то через привилегированную программу сможет получить доступ к запрещенной области.

Биты SUID и GUID нужно использовать аккуратно, и в любом случае владельцем программ не должен быть root или другой привилегированный пользователь. Лучше, если это будет специально заведенная для этой программы учетная запись, которая обладает только теми правами, которые необходимы пользователю.

Рассмотрим еще один пример. Допустим, гость не должен иметь прямого доступа к каталогу **/home/someone**, а для работы программы он необходим. Чтобы не открывать ему лишних возможностей, нужно завести отдельного пользователя с правами доступа на каталог **/home/someone**, сделать его владельцем программы и установить бит SUID. Если в программе будет найдена ошибка, то взломщик получит доступ к **/home/someone**, но все остальные разделы диска останутся недоступными.

Такая политика соответствует нашему основному правилу "Что не разрешено, то запрещено" и обеспечит максимальную безопасность сервера.

4.6. Дополнительные возможности защиты

Помимо прав доступа у любого файла есть еще и атрибуты, которые позволяют построить дополнительную стену безопасности на пути взломщика. Единственное условие — атрибуты могут использоваться только на файловых системах Ext2 и Ext3. Но ограничением это можно назвать с большой натяжкой, потому что Ext3 уже давно становится стандартом для всех дистрибутивов.

Просмотреть текущие атрибуты можно с помощью команды `lsattr`:

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

Первая строка демонстрирует использование команды, а во второй — отображается результат работы. Как видите, мы получили набор символов тире 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 всем содержащимся в нем файлам и подкаталогам, можно использовать ключ `-R`. В этом случае вместо имени файла укажите директорию.

Приведу перечень 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, тем *самым уменьшить количество обращений к диску (избавитесь от лишней операции записи при сохранении файла)*;

- a — позволяет открывать файл только в режиме добавления. Это значит, что уже существующие данные изменить или удалить нельзя;
- d — игнорирует 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 корректировке (изменение, удаление, переименование, создание ссылок);
- s — делает невозможным восстановление файла после удаления. Все пространство на диске, где был файл, 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 запрашивает текущие атрибуты, и в последней строке вы можете увидеть, что в перечне атрибутов первый символ равен "s", а четвертый — "i".

Итак, у нашего файла два взаимоисключающих атрибута. Один запрещает изменения, другой требует полного стирания с диска. Что произойдет, если мы попытаемся удалить файл. Посмотрим?

```
rm test
rm: remove write-protected file "test"?
```

В первой строке мы выполняем команду удаления файла. На это ОС просит подтвердить операцию над защищенным от записи файлом (сообщение показано во второй строке). Как видите, ОС определила наш атрибут i. Попробуйте ввести букву "Y", чтобы удостоверить действие. Вы увидите сообщение об ошибке, и файл останется на месте.

Давайте снимем атрибут i:

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

После отмены атрибута я использовал команду `lsattr` , чтобы убедиться в правильности выполнения команды. Вот теперь файл легко удалить с помощью команды `rm`.

4.7. Защита служб

В данной книге будет рассматриваться множество серверных служб. Безопасность их работы для системы в целом зависит не только от правильной настройки самой службы, но и от прав, которые вы ей дадите. Хакеры очень часто атакуют определенные сервисы и ищут в них изъяны, через которые можно было бы проникнуть в систему, а как мы знаем, ошибки есть всегда и везде.

Во время написания этой книги на мой сайт было произведено несколько удачных атак, потому что в силу занятости я просто не обновлял свой сайт, который располагался на сервере известной хостинговой компании. За два дня на сайте дважды меняли главную страницу, а потом злоумышленники захватили форум. Мне пришлось его убирать в недоступное место, чтобы восстановить свои права администратора, напрямую редактируя базу данных MySQL.

Как произошел взлом? На сайте стоял форум phpBB. Это один из популярных движков и абсолютно бесплатный, поэтому его выбирают многие владельцы сайтов. Большинство хакеров стремятся найти ошибки в наиболее известных разработках, и иногда им это удается. Только своевременное обновление форума (в данном случае) позволяет защититься от нападения.

После атаки я обновил движок, но это не помогло. Разработчики в последнюю версию не включили устранение одной критической ошибки, а инструкции по исправлению дали только на форуме своего сайта. Конечно же, я не увидел этих предписаний и в итоге мог потерять всю базу данных, если бы один из посетителей сайта не указал на ссылку, исправляющую ошибку.

Давайте на примере абстрактного сайта **www.sitename.com** посмотрим, как происходило вторжение. На форуме нужно войти в просмотр какой-нибудь темы, и в строке адреса появится ссылка:

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

Если к этой ссылке добавить в конец текст

&highlight=%2527.$poster=%60**команда Linux**%60.%2527,

то указанная команда Linux выполнится на сервере.

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

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

А вот эта команда могла удалить главную страницу сайта:

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

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

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

После этого случая я целый день бродил по Интернету в поисках уязвимых форумов, и таких оказалось много, потому что администраторы сайтов явно не следят за обновлениями. Я думаю, что в скором времени эти администраторы пройдут через все круги ада. Рано или поздно взломщик найдет форум, и в этом случае нужно 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 раньше хакера, 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 открыт, наверное, на 90 % серверов. Это безграмотность администраторов или их лень? Не знаю, и точно сказать не могу. Только крупные серверы были защищены, а мелкие хостинговые компании явно экономят на заработной плате хороших администраторов.

4.7.1. Принцип работы

Итак, давайте рассмотрим принцип работы защиты служб. Для этого создается директория, которая является для программы корневой. В Linux для этого существует команда `chroot`, которая создает chroot-окружение. Получается псевдокорневая файловая система внутри существующей.

Выше этой директории программа, работающая в окружении chroot, попасть не может. Посмотрите на рис. 4.1. Здесь показана часть файловой системы Linux. Во главе всего стоит корневая директория /. В ней находятся **/bin**, **/usr**, **/var**, **/home**. В папке **/home** расположены каталоги пользователей системы. Мы создаем здесь новую директорию, для примера назовем ее chroot, и она станет корнем для службы. В ней есть свои каталоги **/bin**, **/usr** и т. д., и сервис должен работать с ними, а все, что выше **/home/chroot**, будет недоступно.

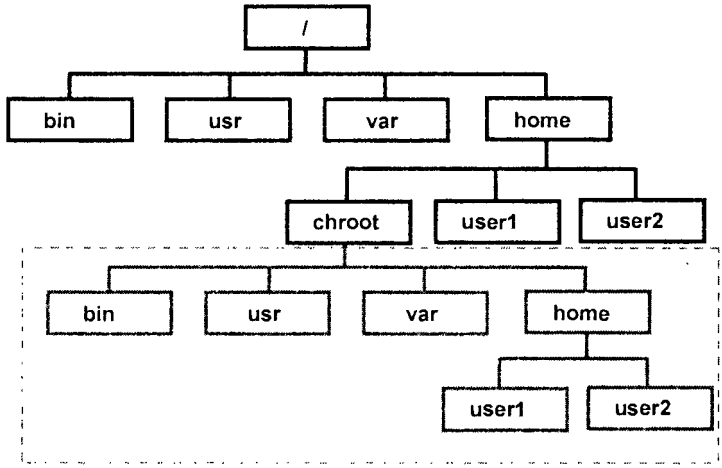

Рис. 4.1. Схема работы chroot-окружения

На рис. 4.1 в рамку обведены папки, которые видны службе. Именно в этом пространстве будет работать сервис, считая, что это и есть реальная файловая система сервера.

Если хакер проникнет в систему через защищенную службу и захочет просмотреть директорию **/etc**, то он увидит каталог **/home/chroot/etc**, но никак не системный **/etc**. Чтобы взломщик ничего не заподозрил, в папке **/home/chroot/etc** можно расположить все необходимые файлы, но содержащие некорректную информацию. Злоумышленник, запросив файл **/etc/passwd** через уязвимый сервис, получит доступ к **/home/chroot/etc/passwd**, потому что служба видит его системным.

Так, например, файл **/home/chroot/etc/passwd** может содержать ложную информацию. На работу системы в целом это не повлияет, потому что ОС будет брать пароли из файла **/etc/passwd**, а службе реальные коды доступа в систему не нужны, поэтому в файл **/home/chroot/etc/passwd** можно засунуть все, что угодно.

4.7.2. Установка jail

Встроенная в Linux-систему программа chroot для создания "Виртуальное пространство" виртуальных пространств на сервере сложна и не очень удобна для применения. Нужно выполнить слишком много операций. Именно поэтому администраторы больше любят использовать 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 разархивировать его, нужно выполнить следующую команду:

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

В текущей директории появится новый каталог **jail** с исходным кодом программы. Да, именно с исходным, потому что она открыта и поставляется в таком виде.

Теперь нужно перейти в каталог **jail/src** (`cd jail/src`) и отредактировать файл Makefile (например, 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 установлен `LINUX`. А следующие три строки для FreeBSD, Irix и Solaris закомментированы. Оставим все, как есть. Что нужно изменить, так это директорию для установки (параметр `INSTALL_DIR`). В последней версии (на момент написания книги) по умолчанию используется каталог **/tmp/jail**. Не знаю, зачем это сделали, ведь этот каталог предназначен для временных файлов и должен быть доступен для чтения абсолютно всем. Раньше по умолчанию был **/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, то воспользуйтесь следующей командой:

```
chmod 755 preinstall.sh
```

Теперь все готово к установке. Находясь в директории **jail/src**, выполните команды:

```
make
make install
```

Если все прошло успешно, то в каталоге **/usr/local/bin** должны появиться программы: addjailsw, addjailuser, jail и mkjailenv.

4.7.3. Работа с программой jail

Для начала создадим каталог **/home/chroot**, который станет корневым для программы, на которой мы будем испытывать систему. Для этого выполним команду:

```
mkdir /home/chroot
```

Теперь нужно подготовить окружение для нормальной работы будущего сервиса. Для этого выполняем команду:

```
/usr/local/bin/mkjailenv /home/chroot
```

Посмотрите, что произошло с каталогом **/home/chroot**. Здесь появились две директории **dev** и **etc**. Как мы знаем, в директории **dev** должны быть описания устройств. В данном случае программа не стала делать полную копию системного каталога **/dev**, а ограничилась созданием трех основных устройств **null**, **urandom** и **zero**.

В директории **etc** можно также увидеть три файла: group, passwd и shadow. Это неполные копии системных файлов. Например, если взглянуть на файл passwd, то он будет содержать только следующие строки:

```
root:x:0:0:Flenov,Admin:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
```

Больше ничего не будет, и нет пользователя robert, которого мы создавали раньше *(см. разд. 4.3)*. В файле shadow находятся теневые пароли. Проверьте права на этот файл, чтобы они были не более 600 (`rw-------`).

Тут есть один недостаток в безопасности — в файле **/home/chroot/etc/shadow** находится реальный зашифрованный пароль из **/etc/shadow**. Лучше удалите его, иначе злоумышленник, узнав пароль на сервис, сможет проникнуть на сервер через другую дверь, которая не защищена виртуальным каталогом.

Продолжаем настройку виртуальной корневой директории. Теперь нам нужно выполнить следующую команду:

```
/usr/local/bin/addjailsw /home/chroot
```

Во время отработки этой команды побежит множество информационных строчек о выполняемых действиях, которые заключаются в том, что в каталог **/home/chroot** копируются основные директории и программы. Например, в папку **/home/chroot/bin** будут скопированы такие программы, как cat, cp, ls, rm и т. д., и сервис будет использовать именно их, а не те, что расположены в основном каталоге **/bin**.

Программа копирует то, что считает нужным, но далеко не все из этого потребуется будущему сервису, который будет работать в виртуальной корневой директории. Лишнее следует удалить, но лучше это делать после того, как убедитесь, что все работает.

Необходимые программы есть и окружение готово. Теперь сюда можно установить сервис:

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

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

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

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

Здесь `chroot` — это виртуальная корневая директория, в нашем случае должно быть **/home/chroot**. Параметр `home` — это домашний каталог пользователя относительно виртуальной директории. Аргументы `sh` — командный интерпретатор и `name` — имя пользователя, которого мы хотим добавить (уже должен 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
```

У меня команда не уместилась в одну строку, поэтому я сделал перенос с помощью символа "\" (обозначает, что директива не закончилась и есть продолжение в следующей строке).

Если параметры указаны верно, то вы должны увидеть уведомление "Done", иначе будет выведено сообщение об ошибке.

Для запуска сервера httpd (в Linux это сервер Apache) в виртуальном окружении должен быть пользователь apache. В реальной оболочке он 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 окажитесь 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**, и в нем будут только пользователи виртуального окружения. Если хакер взломает его, то он получит исключительно эти данные и сможет уничтожить лишь содержимое каталога **/home/chroot**. Вся остальная файловая система останется целой и невредимой.

Для запуска httpd нужно выполнить в виртуальном окружении команду:

```
/usr/sbin/httpd.
```

4.8. Получение прав root

Теперь у нас есть достаточно информации о доступе, и мы можем рассмотреть типичный метод взломщика для получения прав root и способы маскировки в системе.

Допустим, что злоумышленник приобрел возможность выполнять какие-либо системные команды от имени (с правами) root. Сидеть под этой учетной записью будет слишком опасно и вызывающе. К тому же изменять пароль root нельзя.

Как же тогда входить под другим именем и в то же время использовать максимальные права? Давайте вспомним, как Linux работает с правами. В файле **/etc/passwd** хранятся записи пользователей в следующем виде:

```
robert:x:501:501::/home/robert:/bin/bash
```

Третий и четвертый параметры — это идентификаторы пользователя и группы соответственно. Когда на объекты выделяются права, то система сохраняет только идентификаторы. Что это значит? Допустим, что у вас есть пользователь robert с идентификатором 501. Вы создали еще одного пользователя Dan и дали ему такой же идентификатор. Теперь обе учетные записи с одинаковыми ID будут разделять владение одними и теми же объектами.

Что это нам дает? Посмотрите на идентификатор пользователя root — он равен нулю. Именно нулевой ID, а не имя root указывает на максимальные права. Давайте попробуем учетной записи robert, которую мы создали ранее, поставить вручную идентификаторы пользователя и группы, равные 0. В файле **/etc/passwd** должна получиться строка:

```
robert:x:0:0::/home/robert:/bin/bash
```

Теперь войдите в систему под этой учетной записью и попробуйте снова открыть файл **/etc/passwd** и внести изменения или просто добавить пользователя. Все пройдет успешно, хотя файл **/etc/passwd** может изменять только root. Так что система проверяет наши права по идентификатору, который в данном случае нулевой и соответствует максимуму.

Так как имя пользователя ничего не значит, я рекомендую удалить пользователя root в файлах **/etc/passwd** и **/etc/shadow**, создать новую запись с другим именем, но идентификатором, равным 0. Злоумышленники, которые попытаются взломать ваш сервер, будут пытаться подобрать пароль для администратора root, но т. к. пользователя с таким именем нет, то их действия окажутся безуспешными.

С другой стороны, пользователя root можно оставить, но установить ему идентификатор больше нуля. Я иногда создаю учетную запись root с UID=501 или выше. Увидев эту запись, взломщик думает, что он обладает всеми правами, но в реальности оказывается простым пользователем.

Каждая удачная попытка ввести злоумышленника в заблуждение увеличивает вероятность паники со стороны хакера. Даже профессиональный взломщик, находясь в системе, испытывает большую психологическую нагрузку и боится оказаться замеченным. Среди взломщиков немало людей с неустойчивой психикой. Только не думайте, что это сумасшедшие, все хакеры здоровые люди, просто в момент взлома испытывают перенапряжение, и когда что-либо идет не так, как планировалось, хакер может запаниковать.

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 образом в любой момент можно определить идентификатор пользователя и его реальные права.

4.9. Расширение прав

Регламентация доступа — достаточно сложный процесс. Это основная задача администратора и от правильности ее выполнения зависит многое. Любая ошибка может стоить вам зарплаты и благополучия. В мире, когда информация является самым дорогим продуктом, вы должны оберегать ее всеми возможными методами.

Не пожалейте времени и проверьте всю систему на правильность установленных прав. Ни у кого не должно быть ничего лишнего, и в то же время, необходимый доступ должен быть у всех программ для корректной работы.

Честно сказать, права на основе "босс", "друзья босса" и "все остальные" устарели и не обеспечивают необходимой безопасности. Например, у вас есть две группы: бухгалтерия и экономисты. Файлы, созданные любым бухгалтером, будут иметь права `-rwxrwx---` и станут доступными для всех сотрудников этого отдела, потому что в данных правах группа имеет все разрешения на файл, как и владелец.

А как быть, если теперь нужно, чтобы экономист просмотрел документы бухгалтерии? Причем не все пользователи группы экономистов, а только один, и не все файлы бухгалтерии, а выборочно! Решить эту задачу достаточно сложно. Если поставить на файлы бухгалтерии права `-rwxrwxrwx`, то любой пользователь сможет просмотреть бухгалтерскую отчетность, а это уже далеко от идеала безопасности.

Можно пытаться выйти из положения через ссылки или копии файлов с другими правами, но в этом случае вы просто запутаетесь, и дальнейшее управление системой станет затруднительным.

Проблему достаточно просто решить, если ввести списки ACL (Access Control List, списки контроля доступа), как это реализовано в ОС Windows. Сложность только с внедрением, т. к. ОС Linux не имеет стандарта. В принципе, это ядро, на которое любой разработчик может повесить все, что угодно, и каждый производитель ищет свои пути выхода из ситуации или вообще ничего не делает.

Я не могу привести универсальный способ, потому что все решения даются сторонними разработчиками. А это значит, что работоспособность системы можно гарантировать только в отношении определенных версий ядра Linux, которые уже существуют. Нельзя поручиться, что при обновлении версии ядра система списков продолжит функционировать и не вызовет проблем.

Именно поэтому я только предложу взглянуть на проект Linux Extended Attributes and ACLs (**http://acl.bestbits.at/**). Вы можете его использовать только на свой страх и риск.

Linux Extended Attributes and ACLs — продукт, который устанавливается в систему, и после этого требует перекомпиляции ядра. Работа его основана на хранении для каждого файла расширенных атрибутов. Это возможно не на всех файловых системах, поэтому убедитесь, что используемая вами система поддерживает списки ACL. Лучше всего, на мой взгляд, подходят системы ReiserFS и Ext3.

После установки патча и дополнительных программ вам становятся доступны списки ACL. С их помощью можно отдельным пользователям устанавливать режим доступа к файлу. Владельцем файла остается его создатель, и он имеет полные права. Остальные атрибуты доступа могут отсутствовать.

Например, для файла можно установить права `-rwx------`. Несмотря на такие жесткие требования можно указать пользователей, которые будут иметь доступ к этому файлу помимо владельца.

Получается, что кроме основных прав для каждого файла в системе будет храниться список пользователей, которые имеют доступ к нему сверх основной регламентации.

Если бы такой принцип был реализован на уровне ядра и поддерживался всеми дистрибутивами, то я назвал бы ОС Linux самой безопасной и стабильной в мире операционной системой.

4.10. Сетевой экран

Мы достаточно подробно рассмотрели управление доступом к файлам, но на этом распределение прав не заканчивается. Сейчас уже невозможно работать без соединения с локальной сетью или Интернетом, поэтому, прежде чем наш сервер начнет функционировать, нам необходимо ограничить доступ извне к компьютеру и его определенным портам.

Для защиты компьютера от вторжения по сети используется сетевой экран (Firewall). Некоторые службы Linux также имеют свои настройки прав, но их мы будем рассматривать отдельно, когда дойдем до соответствующего сервиса. И все же, я не советую сильно доверять такому управлению. Не забываем, что ошибки есть во всех программах, и если сетевой экран будет дублировать права, прописанные в сервисе, хуже от этого не будет.

Сетевой экран является основой безопасности и первым кольцом защиты от вторжения извне. Хакеру необходимо сначала получить доступ к компьютеру, и только если это удалось, он попытается двигаться дальше и будет про-

биратьcя до уровня файлов. Там уже действует второе кольцо обороны — права доступа к файлам и директориям.

Почему же тогда мы рассматриваем сетевой экран после прав доступа на файлы? Да потому, что Firewall защищает только от сетевых вторжений, а правильная регламентация доступа предохраняет и от локальных хакеров или недобросовестных пользователей, которые получили возможность пользоваться непосредственно терминалом. Оба уровня защиты очень важны. В сфере безопасности вообще нет ничего несущественного, вы должны уделять внимание каждой мелочи.

Сетевой экран позволяет ограничить доступ к компьютеру в целом или к отдельным портам, на которых работают сервисы, но не является 100 % защитой от вторжения. Это всего лишь проверка пакетов на соответствие правилам, которая не может гарантировать, что пользователь является реальным отправителем.

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 и 110 порты соответственно). Я относился к этой категории и не мог получать или отправлять почту, но доступ был у моего начальника. Использование Web-интерфейса для работы с почтовыми сервисами также блокировалось на уровне прокси-сервера. Однажды мне очень срочно нужно было отправить письмо. Для этого я выполнил следующее:

- дождался, когда начальник выйдет из кабинета;
- выключил его компьютер;
- сменил свой IP-адрес на установленный на его компьютере;
- спокойно отправил почту и вернул свой старый IP-адрес.

Когда начальник вернулся, он подумал, что компьютер просто завис, и ничего не заподозрил, а я без проблем смог воспользоваться сервисом, который был запрещен для меня.

Есть множество способов обхода сетевых экранов (не считая ошибок в программах), и все же, правильная конфигурация сможет обеспечить спокойный сон администратора и специалиста по безопасности.

В ОС Linux в качестве Firewall выступает программа, которая фильтрует 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 могут захлебнуться 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 в отдельности (защищать его в зависимости от выполняемых задач) или на входе в сеть (см. рис. 4.2). Во втором случае Firewall реализует общие настройки безопасности для всех компьютеров в сети.

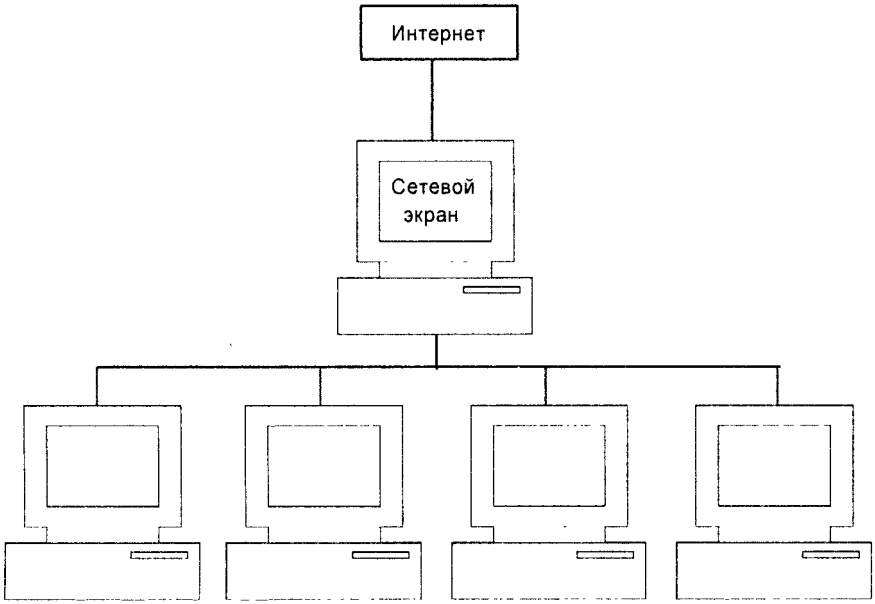

Рис. 4.2. Firewall для защиты сети

Если в вашей сети очень много компьютеров, то управлять ими и обновлять политику безопасности становится затруднительным. Установка единого сервера с Firewall 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 этом случае хакер изначально будет видеть только этот компьютер, а остальные как бы прячутся за занавеской. Чтобы проникнуть на любую машину в сети, злоумышленник должен будет сначала получить доступ к компьютеру с Firewall. Таким образом, защита целой сети упрощается. Подробней о прокси-серверах вы можете узнать в *гл. 9*.

Но во всех сетевых экранах есть одно слабое звено — они программные и используют ресурсы сервера. Современные маршрутизаторы тоже могут решить большинство проблем, которые возлагаются на сетевые экраны Linux. С другой стороны, Linux тоже очень часто используют в качестве маршрутизатора в целях удешевления системы за счет применения старого железа.

4.10.1. Фильтрация пакетов

Итак, основной, но не единственной задачей сетевого экрана является фильтрация пакетов. В Linux уже встроен Firewall, и вам его не надо устанавливать отдельно. Точнее сказать, их даже два: iptables и ipchains. Они позволяют контролировать трафик, который проходит сквозь компьютер по протоколам TCP (Transmission Control Protocol, протокол управления передачей), UDP (User Data Protocol, пользовательский протокол данных) и ICMP (Internet Control Message Protocol, протокол управляющих сообщений в сети Интернет). Так как TCP является транспортом для всех основных протоколов Интернета: FTP (File Transfer Protocol, протокол передачи файлов), HTTP (Hypertext Transfer Protocol, протокол передачи гипертекстовых файлов), POP3 (Post Office Protocol, почтовый протокол) и др., то фильтрация TCP позволяет защищать все эти сервисы.

Все запросы, которые поступают из Интернета или направляются туда, проходят через сетевой экран, который проверяет их по внутренним правилам. И если соответствие установлено, то пакет пропускается. Если какой-либо параметр нарушает хотя бы одно правило, то пакет может быть удален:

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

Я бы не выбирал последний вариант, потому что незачем направлять хакеру лишние пакеты. Лучше оставить действие без внимания, и злоумышленник будет думать, что сервис просто недоступен. Но в этом случае легальные пользователи могут ощутить неудобства при наличии просчета в конфигурировании сетевого экрана. Допустим, что вы по ошибке заблокировали доступ к 80 порту. Если пользователь обратится к Web-серверу, то программа, не получив ответа о запрете, будет находиться в состоянии ожидания до истечения `Timeout`. Для некоторых программ это значение может быть бесконечным, и они зависнут. Исправив ошибку в сетевом экране, вы дадите возможность пользователям работать корректно.

К тому же, отправка сообщений с ошибками идет по протоколу ICMP и увеличивает трафик. Хакер может использовать эти особенности для реализации атаки "Отказ от обслуживания" и переполнить ваш канал ненужными ответами. Атака DoS может быть направлена не только на трафик. Хакеру достаточно в цикле запускать запросы на установку соединения с запрещенным портом, а ваш компьютер будет тратить ресурсы на проверку пакетов и отправку ICMP-ответов. Если пакеты будут идти слишком часто, то сервер может не справиться с нагрузкой и перестанет отвечать на запросы авторизованных пользователей.

При настройке правил можно использовать два варианта фильтра:

1. Разрешено все, что не запрещено.
2. 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 и к обусловленным сервисам. Именно этой политики вы должны придерживаться, когда настраиваете правила для входящих пакетов.

Если двигаться от разрешения, то по умолчанию все позволено. Администратор может забыть или просто не закрыть некий доступ и увидеть свою ошибку только после того, как злоумышленник проникнет в систему.

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 строятся все сервисы — FTP, HTTP, POP3.

Обращаю ваше внимание, что фильтровать можно 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 бессмысленно, но резон есть и достаточно большой. У исходящего трафика могут быть враги, я их перечислю:

- троянские программы, которые могут отправлять в сеть конфиденциальную информацию или соединяться с хакером или с его сервером, чтобы брать команды с какого-нибудь файла;
- специализированные программы для обхода правил. Допустим, что вы запретили доступ к определенному порту извне. Хакер может поместить 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.

Возможных лазеек для проникновения очень много, и ваша задача закрыть максимальное их количество. Для этого под контролем должен быть трафик в обоих направлениях.

Протоколы

TCP используется как базовый для передачи данных таких протоколов, 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 сначала TCP устанавливает соединение с удаленным хостом, и только потом происходит обмен информацией. Благодаря этому подделка IP-адреса любого участника соединения усложняется, а иногда становится и 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 что угодно, и наш сервер не увидит подвоха. Если нет особой надобности, то я запрещаю прохождение таких пакетов в обе стороны.

Протокол ICMP используется 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. Найдите любые способы, чтобы запретить этот протокол. Если обмен управляющими сообщениями необходим в вашей работе, попробуйте найти другую программу, но избавьтесь от использования ICMP.

Фильтрация портов

Первое, на что надо обратить внимание, — это, конечно же, порты. Допустим, что у вас есть Web-сервер, к которому имеют доступ все пользователи. Предположим, что на нем работают абсолютно безопасные сценарии (это фантастика, но допустим ☺), или статичные 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 закачки файлов, ведь бегать с дискетами к Web-серверу никто не будет. Чаще всего для работы с файлами открывают FTP-сервис, а вот это уже дыра.

Для доступа по FTP можно установить наиболее защищенные программы и самые сложные пароли, но хакер рано или поздно сумеет взломать этот сервис. Пароль можно подобрать, украсть с компьютера пользователя или заставить самого сказать через социальную инженерию, 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 будут потрачены все усилия. Да, у одного не получится, у второго, а сотый случайно войдет с первого раза и уничтожит все, что попадется под руку.

Таким образом, можно установить на сервер такую политику, при которой на 80 порт будут приниматься все подключения, а FTP-сервис (21 порт) будет запрещен для всех, кроме определенного IP-адреса. После этого злоумышленник может хоть годами подбирать пароль, любой его трафик будет обрезаться, если он не знает IP-адреса и не сможет его подделать.

Вы должны запретить все порты и после этого открыть только то, что необходимо. На сервере, который охраняет целую сеть, это сделать сложно, потому что разные компьютеры требуют различные сервисы. Открыть их все — значит разрешить работать со всеми портами на любой машине. Конечно же, можно помимо портов использовать в правилах IP-адреса, но дополнительным вариантом защиты будет использование сетевого экрана на каждом компьютере внутри сети. В этом случае каждый из них будет охраняться в зависимости от выполняемых задач. Если это Web-сервер, то из Интернета будет виден только 80 порт, для FTP — 21-й порт.

Фильтрация адресов

Исходя из предыдущих соображений видно, что для фильтрации можно использовать и IP-адрес, хотя максимальный эффект достигается именно в сочетании параметров (порт и адрес).

Допустим, что в вашей сети находятся два Web-сервера. Такое бывает очень часто. Один сервер делают доступным для всех посетителей из Интернета, а второй — только для своих пользователей (внутрикорпоративный сайт). Вполне логичным будет разделение информации, тогда на закрытый сервер можно пускать трафик только локальной сети вне зависимости от порта. Хакеры из Интернета вообще не должны иметь доступ к внутрикорпоративному серверу.

Рассмотрим еще один пример. Допустим, что у вас есть интернет-магазин, который обслуживает заказы пользователей. Вы доставляете товары только по своему городу и не занимаетесь рассылкой. В этом случае нужно разрешить доступ к серверу только с IP-адресов вашего города, а остальным — запретить. Но эта задача достаточно сложна в реализации.

Фильтрация нежелательных адресов

Несколько лет назад сервис **www.regnow.com** (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 Африку и некоторые регионы восточной Европы, включая развивающуюся, но любящую халяву Россию.

Этот шаг оправдан тем, что в некоторых из этих стран очень сильно была развита технология кардинга, когда по ворованным кредитным картам заказывался в Интернете товар. Чтобы кардинг стал недосягаем, сервис запретил доступ по целым группам IP-адресов. Впоследствии выяснилось, что обойти эту систему очень просто. Достаточно воспользоваться анонимным прокси-сервером в США или Канаде, чтобы хакер мог проскочить преграду. А вот у добропорядочных пользователей возникли серьезные проблемы, и они лишились возможности использовать сервис для оплаты необходимых услуг.

Из-за серьезных недостатков данный фильтр был изъят, 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 использования.

Фильтрация неверных адресов

Был случай, когда один сервис неправильно обрабатывал адрес получателя. Если серверу приходил неверный пакет, то он отвечал отправителю сообщением о некорректности данных. Проблема заключалась в том, что злоумышленник мог послать на сервер такой пакет, в котором в качестве отправителя стоял адрес получателя, т. е. и в том и в другом случае использовался IP-адрес сервера. Конечно же, сервис пытался отослать сообщение об ошибке, и отправлял его сам себе, и снова видел ошибочный пакет. Таким образом информация зацикливалась. Если злоумышленник направит тысячи таких пакетов, то сервер только и будет посылать сообщения об ошибках.

О подобных погрешностях я уже давно ничего не слышал, но нет гарантии, что они не появятся снова. Существует множество адресов, которые надо фильтровать и не пропускать в сеть.

Помимо этого, я советую не пропускать пакеты с адресами, которые зарезервированы или не могут использоваться в Интернете. Рассмотрим диапазоны этих адресов:

- в качестве отправителя стоит адрес 127.0.0.1. Из Интернета пакет с таким адресом прийти не может, потому что он всегда используется для указания на локальную машину (localhost);
- от 10.0.0.0 до 10.255.255.255 — используется для частных сетей;
- от 172.16.0.0 до 172.31.255.255 — выделен для частных сетей;
- от 192.168.0.0 до 192.168.255.255 — зарезервирован для частных сетей;

- от 224.0.0.0 до 239.255.255.255 — используется для широковещательных адресов, которые не назначаются компьютерам, поэтому с них не могут приходить пакеты;
- от 240.0.0.0 до 247.255.255.255 — зарезервирован для будущего использования в Интернете.

Все эти адреса нереальны для Интернета, и никакие пакеты с такими параметрами не должны проходить через сетевой экран.

Фильтрация в Linux

Для фильтрации пакетов по определенным вами правилам в ядро Linux уже встроены все необходимые функции. Но это только основа, а нужен еще инструмент, который в удобной форме позволит управлять этими правилами.

В ОС Linux включены сразу две программы: iptables и ipchains (вызываются одноименными командами). Какая из них лучше, — сказать сложно, потому что они схожи по своим возможностям. Но многие профессионалы останавливаются на ipchains. Выбор остается за вами. Одни любят старые и проверенные методы, а другие предпочитают все новое.

В ядре Linux находятся три основные цепочки (chain) правил:

- Input — для входящих пакетов;
- Output — для исходящих пакетов;
- Forward — для пакетов, предназначенных другой системе.

Вы можете создавать свои цепочки, которые будут привязаны к определенной политике, но мы эту тему рассматривать не будем.

ОС Linux проверяет все правила из цепочки, которая выбирается в зависимости от направления передачи. Пакет последовательно обследуется на соответствие каждому правилу из цепочки. Если найдено хотя бы одно совпадение с описанием, то выполняются действия, указанные в данном правиле: DENY, REJECT или ACCEPT, т. е. система решает, пропускать пакет дальше или нет.

Цепочки несут в себе одну очень неприятную для новичков особенность. Допустим, что на вашем сервере вы хотите открыть 21 порт только для себя. Для этого можно создать два правила:

1. Запретить все входящие пакеты на 21 порт сервера.
2. Разрешить пакеты на 21 порт с компьютера с адресом 192.168.1.1.

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 сравнение первого правила с параметрами пакета даст поло-

жительный результат, т. е. выполнится запрет и пакет удалится, и второе правило не сработает никогда.

Чтобы наша политика действовала верно, строки надо поменять местами. В этом случае сначала проверится запись "Разрешить пакеты на 21 порт с компьютера с адресом 192.168.1.1", контроль пройдет успешно и пакет будет пропущен. Для остальных IP-адресов это правило не выполнится, и проверка продолжится. И вот тогда сработает запрет доступа на 21 порт для всех пакетов.

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

4.10.3. Firewall — не панацея

Установив Firewall, вы не получаете максимальной защиты, потому что существует множество вариантов обхода не только конкретных реализаций, а любого сетевого экрана.

Firewall — это всего лишь замок на двери парадного входа. Злоумышленник никогда не воспользуется парадным входом, он будет проникать в систему через черный или полезет в окно. Например, на рис. 4.2 показана защищенная сеть, а ее главная дверь — это выход в Интернет через сетевой кабель. А если на каком-нибудь клиентском компьютере стоит модем, то это уже черный ход, который не будет контролироваться сетевым экраном.

Я видел серверы, в которых доступ в сеть разрешен только с IP-адресов, определенных списком. Администраторы верят, что такое правило позволит защититься от хакеров. Это не так, потому что IP-адрес легко подделать.

Однажды я работал на фирме, где выход в Интернет контролировался по IP-адресу. У меня было ограничение на получение *100 Мбайт информации в месяц*, а на соседнем компьютере — полный доступ. Чтобы не тратить свой трафик на получение больших файлов со своего IP-адреса, я только смотрел Web-страницы. Когда нужно было что-либо скачать, я выполнял следующие действия:

- дожидался, когда освободится нужный компьютер, например, когда хозяин уходил на обед;
- вытаскивал сетевой кабель на компьютере соседа, чтобы разорвать соединение;
- спокойно менял свой IP-адрес на соседский, и по безграничному трафику быстро качал все, что требовалось;
- после скачивания возвращал IP-адрес и кабель на место.

Таким способом я в течение месяца получал необходимую мне информацию.

Дальше стало еще проще. Я установил прокси-сервер на соседний компьютер и использовал его. После этого мы всем отделом заходили в Интернет через один IP-адрес, имеющий неограниченный трафик.

В современных сетевых экранах простая замена IP-адреса не позволить извне проникнуть в систему. Сейчас используются намного более сложные методы идентификации. С помощью подмены можно получить большие привилегии только в рамках сети, а не извне, да и то лишь при плохих настройках. Но хороший администратор даже внутри сети не допустит таких махинаций, потому что есть еще защита по MAC-адресу и пароли доступа.

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, которые есть всегда и везде.

Рассмотрим защиту по портам. Допустим, что у вас есть Web-сервер, который охраняется сетевым экраном с разрешенным только 80 портом. А ему больше и не надо!!! Но 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 на выполнение на сервер через ICMP-сообщения Echo Request (эхо-запрос) и Echo Reply (эхо-ответ), подобно команде `ping`.

Сетевой экран помогает защищать данные, но основным блюстителем порядка является администратор, который должен постоянно следить за безопасностью и выявлять атаки. Вновь разработанная атака сможет преодолеть сетевой экран, и компьютер ничего не заподозрит, потому что определяет только заложенные в программу алгоритмы. Чтобы обработать нестандартную ситуацию, за системой должен наблюдать администратор, который будет реагировать на любые нештатные изменения основных параметров.

Для того чтобы пройти через сетевой экран, зачастую требуется пароль или предъявление какого-нибудь устройства типа Touch Memory, Smart Card и др. Если пароль не защищен, то все затраченные на сетевой экран деньги окажутся потерянными зря. Хакер может подсмотреть пароль или подслушать его с помощью анализатора пакетов (сниффер) и предъявить подделанные параметры идентификации сетевому экрану. Таким образом было взломано немало систем.

Управление паролями должно быть четко определено. Вы должны контролировать каждую учетную запись. Например, если уволился сотрудник, у которого были большие привилегии, то все сведения, определяющие его в ОС, необходимо тут же заблокировать и изменить все известные ему пароли.

Однажды мне пришлось восстанавливать данные на сервере после того, как фирма выгнала администратора. Он посчитал увольнение несправедливым и через несколько дней без особого труда уничтожил на главном компьютере всю информацию. Не спас даже хорошо настроенный сетевой экран. В данном случае все произошло быстро, потому что взломщик сам занимался установкой параметров. Такого не должно быть, необходимо конфигурировать Firewall так, чтобы его не взломал даже бывший администратор.

Я всегда рекомендую своим клиентам, чтобы пароль с основными привилегиями на сетевой экран был известен только одному человеку, например, начальнику информационного отдела, но никак не рядовому специалисту. Администраторы меняются достаточно часто, и после каждого их увольнения можно забыть поменять какой-нибудь пароль.

4.10.4. Firewall как панацея

Может сложиться впечатление, что Firewall — это пустое развлечение и трата денег. Это не так. Если он хорошо настроен, 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. Да, и MAC и IP-адрес легко подделать, но можно для полной надежности подключить к системе защиты и порты на коммутаторе. В этом случае, даже если хакер будет знать пароль, то для его использования нужно сидеть именно за тем компьютером, за которым он закреплен. А это уже не просто.

Защита может и должна быть многоуровневой. Если у вас есть данные, которые нужно оградить от посторонних, то используйте максимальное количество уровней. Помните, что лишней защиты не бывает.

Представьте себе банк. У входа обязательно стоит секьюрити, который спасет от воров и мелких грабителей. Но если подъехать к такой организации на танке, то такие блюстители порядка не помогут.

Сетевой экран — это как охрана на дверях, защищает от мелких хакеров, которых подавляющее большинство. Но если банком займется профессионал, то он может добиться успеха.

Помимо охраны у входа, деньги в банке всегда хранятся в сейфе. Финансовые сбережения для банка — это как секретная информация для сервера, и они должны быть максимально защищены. Именно поэтому устанавливают сей-

фы со сложными механизмами защиты, и если не знать, как их обойти, вор потратит на вскрытие замка драгоценное время, и успеет приехать милиция.

В случае с сервером в роли сейфа может выступать шифрование, которое повышает гарантию сохранности данных. Даже если хакер проникнет на сервер, минуя сетевой экран, ему понадобится слишком много времени, чтобы расшифровать данные. Вы успеете заметить и вычислить злоумышленника. Ну а если взломщик скачал зашифрованные данные и пытается их раскодировать на своем компьютере, то с большой вероятностью можно утверждать, что информация раньше устареет, чем хакер сможет ее прочитать. Главное, чтобы алгоритм шифрования и ключ были максимально сложными.

4.10.5. Конфигурирование Firewall

Самый простой способ настроить сетевой экран — использовать графическую утилиту. Для этого загрузитесь в графическую оболочку и выберите из главного меню KDE строку **System\Firewall Configuration**. Перед вами откроется окно из двух вкладок: **Rules** и **Options**.

Первым делом советую перейти на вкладку **Options**. Она показана на рис. 4.3. Здесь можно указать действия по умолчанию для каждого типа пакетов (Input, Forward и Output) и ограничить прохождение пакетов ICMP.

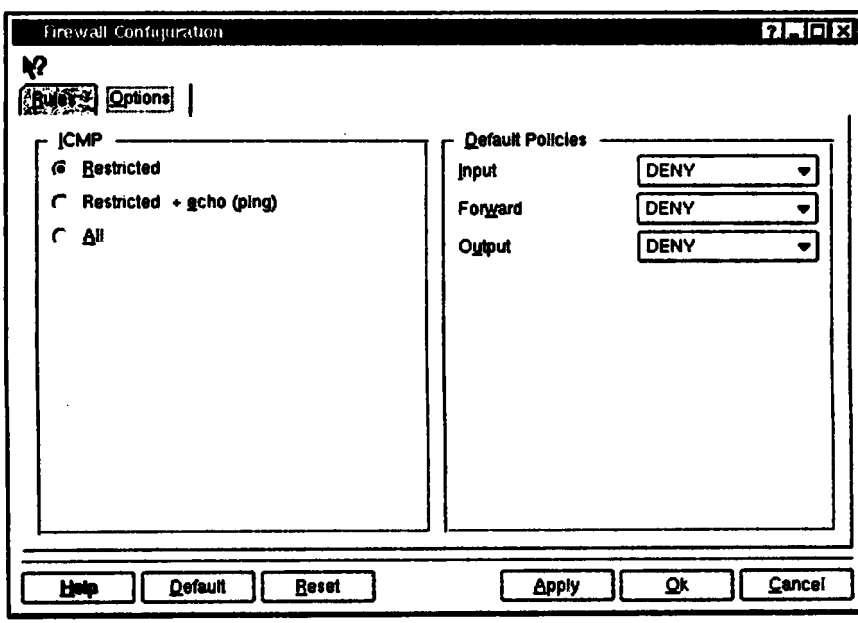

Рис. 4.3. Вкладка **Options** программы конфигурирования Firewall

После этого переходим на вкладку **Rules** (рис. 4.4), где можно добавлять, изменять или удалять правила фильтрации. Попробуйте нажать на кнопку **New**, чтобы создать новое правило. Перед вами откроется окно, как на рис. 4.5.

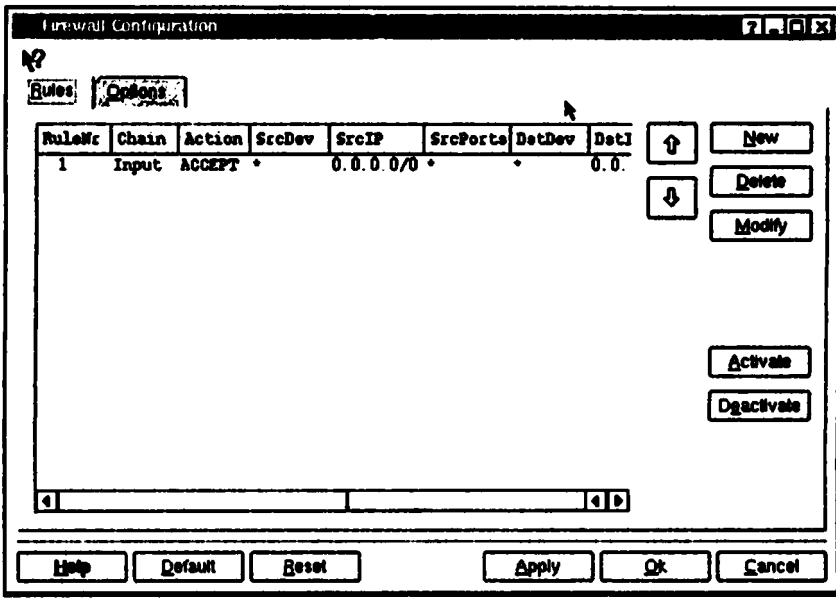

Рис. 4.4. Вкладка Rules программы конфигурирования Firewall

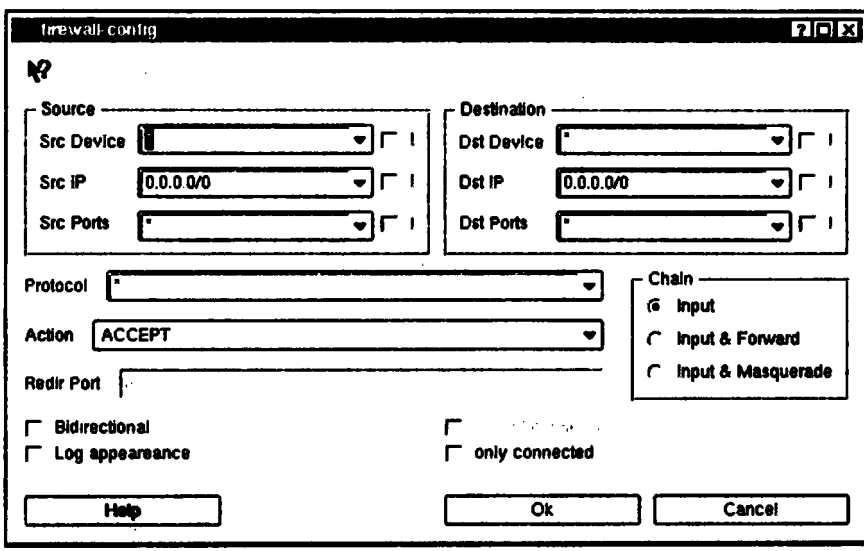

Рис. 4.5. Окно добавления нового правила

Пока что не надо ничего изменять. Если вы загрузили сейчас утилиту Firewall Configuration, то советую только посмотреть на ее внешний вид и предоставляемые возможности. Настройками можно будет заняться тогда, когда вы познакомитесь с программой ipchains, которая из командной строки позволяет изменять параметры сетевого экрана. Для этого мы рассмотрим множество интересных и полезных на практике примеров, которые помогут вам разобраться и с конфигурированием Firewall.

4.11. ipchains

Наиболее распространенной программой для создания правил сетевого экрана является ipchains. В команде вызова можно указывать следующие параметры:

- `-A цепочка правило` — добавить правило в конец цепочки. В качестве аргумента указывается имя цепочки `input`, `output` или `forward`;
- `-D цепочка номер` — удалить правило с указанным номером из заданной цепочки;
- `-R цепочка номер правило` — заменить правило с указанным номером в заданной 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 указанной цепочки;
- `-N имя` — создать новую цепочку с заданным именем;
- `-X имя` — удалить цепочку с указанным именем;
- `-P цепочка правило` — позволяет изменить политику по умолчанию;
- `-p протокол` — определяет протокол, к которому относится правило. Значений аргумента `протокол` может быть четыре: `tcp`, `udp`, `icmp` или `all` (указывается, если правило действует для всех протоколов);
- `-i интерфейс` — сетевой интерфейс, к которому будет привязано правило. Если этот аргумент не указан, то правило будет относиться ко всем интерфейсам;
- `-j действие` — операция, которая должна быть выполнена над пакетом. В качестве аргумента можно указать `ACCEPT`, `REJECT` или `DENY`;
- `-s адрес порт` — характеристики отправителя пакета. Первый аргумент задает IP-адрес источника, а второй (не обязательный) — порт. Будьте внимательны, у протокола ICMP нет портов;

- `-d адрес порт` — атрибуты получателя пакета. Первый аргумент задает IP-адрес назначения, а второй (не обязательный) — порт.

4.11.1. Фильтр по умолчанию

Исходя из принципа всеобщего запрета, в качестве правила по умолчанию мы должны запретить любые действия. Изначально в Linux все разрешено, а это безопасно только для отдельно стоящего сервера, который даже не подключен к сети. Проверьте ваши настройки по умолчанию, выполнив команду:

```
ipchains -L
```

В результате вы должны увидеть на экране примерно следующее:

```
Chain input (policy ACCEPT):
Chain forward (policy ACCEPT):
Chain output (policy ACCEPT):
Chain icmp (0 references):
```

В зависимости от дистрибутива настройки по умолчанию могут отсутствовать, тогда вместо правил вы увидите следующее сообщение об ошибке:

`ipchains: Incompatible with this kernel` (невозможно для этого ядра).

Такая фраза может появиться, если ipchains отсутствует или запущена неверно. Я уже не раз встречал это сообщение, потому что производители дистрибутивов неверно конфигурируют систему по умолчанию. А ведь лечится эта ошибка достаточно просто, и ядро тут ни при чем.

Просмотрите файл **/etc/rc.d/init.d/ipchains** с помощью текстового редактора программы MC или команды `cat`. Первое будет удобнее. Нужно найти в файле следующую строку:

```
IPCHAINS_CONFIG=/etc/sysconfig/ipchains
```

Путь к файлу в директиве `IPCHAINS_CONFIG` может быть другим В современных дистрибутивах в директории **/etc/sysconfig/** хранятся файлы конфигурации служб. Для сервиса ipchains это файл ipchains. Проверьте его существование следующей командой:

```
ls /etc/sysconfig/ipchains
```

Если файл не существует, то его следует создать. Для этого выполним следующую команду:

```
cat >> /etc/sysconfig/ipchains
```

Отныне все команды, вводимые в консоль, будут сохраняться в файле. Для того чтобы сервис ipchains заставить работать, достаточно ввести:

```
:input ACCEPT
```

Теперь нажмите клавиши <Ctrl>+<D> и перезапустите сервис ipchains следующей командой:

```
/etc/rc.d/init.d/ipchains restart
```

Обратите внимание, что нужно указать полный путь. Если этого не сделать, то будет запущена утилита ipchains, а не сценарий запуска из директории **/etc/rc.d/init.d/**.

Вот теперь команда должна выполниться успешно.

Для начала давайте запретим любой трафик. Но прежде, чем мы перейдем к созданию правил, сделаю еще одно замечание. Любую систему нужно начинать конфигурировать с чистого листа, потому что настройки по умолчанию могут оказаться не слишком эффективными и небезопасными. Выполните команду `ipchains -F`, чтобы обнулить текущие настройки. Сетевой экран — это сердце защиты, поэтому в данном случае очистка является наиболее важной.

Теперь зададим политику по умолчанию. Для этого нужно выполнить команду `ipchains` с параметром `-P` для каждой цепочки и указать политику безопасности:

```
ipchains -P input DENY
ipchains -P output REJECT
ipchains -P forward DENY
```

Обратите внимание, что для входящих (input) и проходящих (forward) пакетов я установил полный запрет, поэтому они будут удаляться без каких-либо предупреждений. Для исходящей информации можно поставить в правило `REJECT`, чтобы внутренние клиенты при попытке подключения к серверу могли получить сообщение об ошибке.

Теперь ваш компьютер невиден и недоступен в сети. Попробуйте просканировать порты или выполнить команду `ping` к серверу. Оба действия не дадут результата, как будто компьютера вообще не существует для окружающих.

4.11.2. Примеры добавления ipchains-правил

Давайте теперь указывать права, чтобы разрешить какой-либо доступ к серверу. Только вы должны учитывать, что если добавлять правило в конец набора, то нет гарантии, что оно будет работать верно. В цепочке уже может быть запрет, поэтому доступ будет закрыт раньше, чем сработает наше правило. Чтобы не столкнуться с такой проблемой, я в примерах буду вставлять новое правило первым (указывать ключ `-I` и номер `1`).

Если вы добавляете общее правило запрета, то его следует поместить в самый конец. Когда правило касается конкретного действия, порта или адреса, то

его место в начале цепочки. Таким образом, в вашем своде сначала будут идти специфические правила, а потом глобальные.

Допустим, что у нас используется публичный Web-сервер, и нужно разрешить всем пользователям работать с 80 портом (именно его используют Web-серверы по умолчанию). Для решения этой проблемы выполняем команды:

```
ipchains -I input 1 -p tcp -d 192.168.77.1 80 -j ACCEPT
ipchains -I output 1 -p tcp -s 192.168.77.1 80 -j ACCEPT
```

В качестве порта можно указывать как числовое значение, так и имя. Это значит, что приведенные выше команды можно записать таким образом:

```
ipchains -I input 1 -p tcp -d 192.168.77.1 web -j ACCEPT
ipchains -I output 1 -p tcp -s 192.168.77.1 web -j ACCEPT
```

Здесь вместо 80 порта указано его имя `web`, и программа ipchains корректно отработает с этим аргументом.

Рассмотрим каждый ключ первой строки в отдельности:

- `-I input 1` — ключ `-I` указывает на необходимость вставки правила под заданным номером. Потом показываем цепочку, в которую надо вставить правило, в данном случае это input. Цифра `1` обозначает номер, под которым вставляется правило, т. е. оно будет первым;
- `-p tcp` — Web-сервер работает по протоколу HTTP, который использует в качестве транспорта TCP. С помощью ключа `-p` мы в явном виде указываем протокол. Не забывайте делать это. В противном случае вы откроете одновременно доступ к сервисам на двух портах (TCP и UDP). Хорошо, если на 80 порту UDP в этот момент не будет работать какая-нибудь программа;
- `-d 192.168.77.1 80` — правило проверяет, чтобы получателем был порт 80 (или его имя web) на сервере с адресом 192.168.77.1. В данном случае этот IP принадлежит моему серверу. Это значит, что я разрешил входящие пакеты на 80 порт своего компьютера. Адрес отправителя в правиле не указан, а значит, может быть любым;
- `-j ACCEPT` — опция устанавливает разрешение. Если пакет соответствует правилам, заданным предыдущими ключами (в данном случае проверяется адрес и порт назначения и протокол), то он будет пропущен.

В соответствии с первой строкой посылать на сервер запросы разрешается всем. Но Web-сервер должен иметь возможность возвращать страницы на запросы клиентов. Для этого нужно открыть 80 порт моего сервера (192.168.77.1) для всех исходящих пакетов. Именно это делает вторая строка.

Выполните команду `ipchains -L`, и в результате вы должны увидеть следующее содержимое всех ваших цепочек:

```
Chain input (policy DENY):
target    prot opt    source      destination    ports
ACCEPT    tcp ------ anywhere     flenovm.ru     any -> http

Chain forward (policy DENY):

Chain output (policy DENY):
target    prot opt    source      destination    ports
ACCEPT    tcp ------ flenovm.ru   anywhere       http -> any
Chain icmp (0 references):
```

В цепочках input и output появилось по одной строке. Обратите внимание, что в колонках `source` и `destination` обеих цепочек IP-адрес заменился на доменное имя моего компьютера `flenovm.ru`. Если сервер может определить имя, то он делает такую подмену. Посмотрите на колонку `ports`, здесь номер порта 80 заменен на http.

Я советую вам внимательно проанализировать созданный нами список фильтров, чтобы вы четко понимали каждую его колонку. Рассмотрим структуру строк на примере цепочки input:

```
target    prot opt    source      destination    ports
ACCEPT    tcp ------ anywhere     flenovm.ru     any -> http
```

Первая строка — имена столбцов, а вторая — это фильтр, содержащий реальные значения. Здесь у нас 6 колонок:

- `target` — действие, которое будет выполняться, если пакет удовлетворяет фильтру. В нашем случае стоит `ACCEPT`, т. е. пропустить дальше, в противном случае пакет уничтожается;
- `prot` — протокол, в данном случае `tcp`;
- `opt` — дополнительные опции. Мы их не указывали, поэтому здесь стоят прочерки;
- `source` — источник пакета. Слово "anywhere" указывает на то, что посылка может быть от любого компьютера;
- `destination` — адресат. Здесь может быть имя компьютера или его IP-адрес;
- `ports` — порт, указывается в виде `источник -> назначение`. В данном случае у источника может быть любой порт (используется слово `any`), а пункт назначения должен работать только через http (80 порт).

Как это часто бывает, Web-сервер должен кто-то обновлять, и обычно это делается через FTP-сервис. Всем доступ открывать нельзя, поэтому пропишем правило, по которому подключаться к FTP-серверу (21 порт) сможет только один компьютер с адресом 192.168.77.10. Для этого выполняем следующие команды:

```
ipchains -I input 1 -p tcp -d 192.168.77.1 21 \
  -s 192.168.77.10 -j ACCEPT
ipchains -I output 1 -p tcp -s 192.168.77.1 21 \
  -d 192.168.77.10 -j ACCEPT
```

В данном примере пропускаются пакеты, входящие на 21 порт сервера с адресом 192.168.77.1 с любого порта компьютера 192.168.77.10. Вторая строка разрешает исходящие пакеты с 21 порта сервера 192.168.77.1 на компьютер клиента с адресом 192.168.77.10.

Если у вас настроен FTP-сервер и вы сейчас к нему подключитесь, то не увидите файлов и не сможете работать. В отличие от Web-сервера протокол FTP требует для работы два порта: 21 (ftp-порт для передачи команд) и 20 (ftp-data порт для обмена данными). Поэтому нужно открыть доступ и к 20 порту:

```
ipchains -I input 1 -p tcp -d 192.168.77.1 20 \
  -s 192.168.77.10 -j ACCEPT
ipchains -I output 1 -p tcp -s 192.168.77.1 20 \
  -d 192.168.77.10 -j ACCEPT
```

Теперь компьютер с адресом 192.168.77.10 имеет полноценный доступ к FTP-сервису, а для всех остальных он недоступен. Сканирование сервера с любого компьютера вашей сети покажет открытым только 80 порт, и лишь с компьютера с IP 192.168.77.10 можно будет увидеть 21 и 80 порты.

Выполните команду `ipchains -L`, чтобы просмотреть текущее состояние ваших цепочек. Вы должны увидеть примерно следующий результат:

```
Chain input (policy DENY):
target  prot opt  source     Destination  ports
ACCEPT  tcp ------ 192.168.77.10  flenovm.ru  any -> ftp-data
ACCEPT  tcp ------ 192.168.77.10  flenovm.ru  any -> ftp
ACCEPT  tcp ------ anywhere    flenovm.ru  any -> http

Chain forward (policy DENY):

Chain output (policy DENY):
target  prot opt   source     destination  ports
ACCEPT  tcp ------ flenovm.ru   192.168.77.10  ftp-data -> any
ACCEPT  tcp ------ flenovm.ru   192.168.77.10  ftp -> any
```

```
ACCEPT  tcp ------ flenovm.ru   anywhere    http -> any
Chain icmp (0 references):
```

Через фильтры, описанные в этом разделе, пропускаются любые пакеты, вне зависимости от интерфейса. В большинстве случаев это оправдано, но петля (loopback, всегда указывает на вашу машину), чаще всего, не нуждается в защите. Ее можно использовать только локально, и ни один хакер не сможет подключиться через этот виртуальный интерфейс удаленно. Вполне логичным будет разрешить все пакеты через loopback:

```
ipchains -A input -i lo -j ACCEPT
ipchains -A output -i lo -j ACCEPT
```

Большинство администраторов не любят открывать полный доступ через loopback, потому что политики внешнего и виртуального интерфейсов будут различаться. В этом случае тестирование сетевых программ усложняется. Проверив программу через lo, нет гарантии, что она будет функционировать с удаленными подключениями, ведь там могут помешать нормальной работе фильтры сетевого экрана.

4.11.3. Примеры удаления ipchains-правил

Попробуем отменить доступ к FTP на примере удаления записей из цепочки input. Я специально выбрал в качестве образца FTP-сервис, потому что он требует две строки описания, и при совершении операции нужно быть очень внимательным. На вскидку нужно выполнить следующие команды:

```
ipchains -D input 1
ipchains -D input 2
```

Пока не спешите это делать. Ключ -D свидетельствует о необходимости удаления правила. После него указана цепочка, с которой надо произвести операцию, и номер записи. Обратите внимание на последовательность удаления в данном примере (сначала 1, а потом 2 запись). Ничего не заметили?

Если выполнить первую строку и потом просмотреть содержимое цепочки input, то в результате мы получим:

```
Chain input (policy DENY):
target  prot opt  source     Destination  ports
ACCEPT  tcp ------ 192.168.77.10  flenovm.ru  any -> ftp
ACCEPT  tcp ------ anywhere    flenovm.ru  any -> http
```

Строка для порта ftp-data отсутствует, но остальные записи сместились, и при выполнении команды ipchains -D input 2 мы удалим разрешение для http-сервера, а доступ к ftp-порту останется. Это можно пережить, когда записей

только три, а что если их будет сотня? Вспомнить, какая строка была удалена, очень сложно.

Чтобы не столкнуться с такой ситуацией, удаление начинайте с последней строки. В данном случае команды надо расположить таким образом:

```
ipchains -D input 2
ipchains -D input 1
```

Есть еще один способ удаления, который понадежнее, но для его рассмотрения давайте создадим отдельную запись в цепочке forward. Выполните следующую команду:

```
ipchains -A forward -p icmp -j DENY
```

В этой строке мы используем ключ `-A`, который добавляет строку в конец цепочки (в нашем случае была пустая).

Внимание!

Если в вашей системе запрещена переадресация (forward), то запись будет добавлена, но на экране может появиться предупреждение. Чуть позже в *разд. 4.11.7* мы поговорим о перенаправлении более подробно.

Рассмотрим, что делает это правило. Оно сработает, если пакет требует перенаправления и при этом использует ICMP-протокол. Мы установили фильтр `DENY`, а значит, пакет будет просто удален. Таким образом, мы заблокировали ICMP-трафик для перенаправления. Чтобы запретить ICMP-пакеты вообще, нужно добавить еще правило:

```
ipchains -A input -p icmp -j DENY
```

Теперь попробуем удалить запись. Для этого наберите команду, которую вы выполняли для добавления правила, но замените ключ `-A` (или `-I`, если вы использовали вставку) на `-D`. В результате должна получиться команда:

```
ipchains -D forward -p icmp -j DENY
```

Выполните ее и убедитесь, что запись удалена успешно.

4.11.4. Правила "все кроме"

Очень часто приходится задавать правила в виде "все кроме". Например, нужно запретить доступ к порту telnet всем пользователям, кроме компьютера с адресом 192.168.77.10. Лучше поступить следующим образом: сначала разрешить доступ для компьютера 192.168.77.10, а потом запретить всем. Тогда в цепочке input будет две записи:

- разрешить подключение к telnet с адреса 192.168.77.10;
- запретить подключение к telnet.

Мы исходили из того, что по умолчанию разрешено все. Тогда, как и положено, все входящие пакеты будут сверяться с первой строкой, и если пакет пришел не с адреса 192.168.77.10, то вторая запись удалит его.

А ведь можно проблему решить одной строкой. Для этого нужно использовать следующее правило:

```
ipchains -I input 1 -p tcp -s ! 192.168.77.10 telnet -j DENY
```

В данной команде мы запрещаем (ключ `-j DENY`) подключение по протоколу TCP (`-p tcp`) всем пакетам, у которых адрес источника (ключ `-s`) не равен `192.168.77.10`. Символ "!" выступает как знак неравенства, т. е. фильтру будут соответствовать все пакеты, в которых источник не равен указанному.

Такая запись эффективна, если по умолчанию все разрешено. В противном случае пакет от компьютера `192.168.77.10` будет все равно удален.

Знак "!" можно использовать и перед указанием номера порта. Например, нужно разрешить с адреса 192.168.77.12 полный доступ к серверу, кроме порта telnet. Тогда по умолчанию делаем запрет всего и открываем доступ только компьютеру 192.168.77.12:

```
ipchains -I input 1 -p tcp -s 192.168.77.12 ! telnet -j ACCEPT
```

Данная строка разрешает подключение по протоколу TCP компьютеру с адресом 192.168.77.12 на любые порты за исключением telnet, потому что перед именем порта стоит восклицательный знак.

4.11.5. Ограничение сети

В крупных сетях описывать каждый компьютер очень сложно. Для облегчения этой задачи можно использовать групповые записи. Например, вам надо разрешить выход в Интернет только для сети 192.168.1.x (с маской 255.255.255.0). Это значит, что первые 24 бита (первые три октета) адреса соответствуют идентификатору сети, а последние 8 бит (последнее число) — это номер компьютера в ней. Чтобы разрешить доступ для всей сети, можно выполнить команду:

```
ipchains -I input 1 -p tcp -s 192.168.1.0/24 -j ACCEPT
```

В данном случае в качестве адреса указано значение 192.168.1.0/24. После слэша как раз идет количество бит, которые определяют адрес сети. Это значит, что все компьютеры в этой сети попадают под данный фильтр.

Существуют три основных класса сетей, которые отличаются количеством бит, отведенных под адрес. Чтобы вам было проще, приведу эту классификацию:

- A — первые 8 бит определяют адрес сети. В такой сети используются адреса в диапазоне 01.0.0.0 до 126.0.0.0;

- B — первые 12 бит определяют адрес сети. В такой сети используются адреса в диапазоне от 128.0.0.0 до 191.255.0.0;
- C — первые 16 бит определяют адрес сети. В такой сети используются адреса в диапазоне от 192.0.1.0 до 223.255.255.0.

Существуют и исключения, которые мы рассматривали в *разд. 4.10.2*. Если вы не сталкивались раньше с TCP-протоколом, то я советую познакомиться с ним сейчас. Это поможет вам в администрировании вашей системы.

4.11.6. ICMP-трафик

Самым сложным для многих администраторов является управляющий протокол ICMP, который по стандарту RFC 792 требуется для работы протокола TCP/IP. Но в жизни не всегда придерживаются стандартов, и TCP/IP может работать на компьютерах, где ICMP запрещен.

Протокол TCP жестко прошивают в наши головы, и с ним сталкиваются любые пользователи, потому что TCP наиболее распространен. Протокол UDP по своим характеристикам схож с TCP, поэтому тут тоже особых проблем не возникает. А вот ICMP мало кому знаком и его даже бояться. Бытует мнение, что его проще и лучше полностью запретить, но это не так.

Протокол ICMP позволяет двум узлам в сети совместно использовать информацию об ошибках. Проходящие через него пакеты предназначены не определенной программе, а компьютеру в целом, поэтому у него нет портов. Зато есть тип и код. Эти параметры протокола можно увидеть, если выполнить команду:

```
ipchains -h icmp
```

Пакеты ICMP, чаще всего, рассылаются в ответ на нештатные ситуации. Основные типы и коды, которые доступны для управления, можно увидеть в табл. 4.1.

При создании правила тип ICMP-сообщения указывается так же, как и порт для TCP-протокола, а код — помещается после ключа `-d`. Например, следующая строка запретит ICMP-пакеты третьего типа с кодом 1:

```
ipchains -I output 1 -p icmp -s 192.168.8.1 3 -d 1 -j DENY
```

Некоторые администраторы уделяют недостаточное внимание ICMP-протоколу. Это ошибка, потому что таким путем было совершено уже достаточно много атак, к тому же хакер может с помощью сообщений ICMP передавать даже TCP-трафик, используя туннелирование.

Таблица 4.1. Основные коды и типы ICMP-пакетов

Тип	Код	Описание
0	0	echo-reply — такие пакеты используются для проверки связи с компьютером через утилиту ping
3	0—7	destination-unreachable — пакеты этого типа указывают на недоступность адресата. В зависимости от кода можно получить уточненные данные: • 0 — недоступна сеть; • 1 — недоступен компьютер; • 2 — недоступен протокол; • 3 — недоступен порт; • 4 — требуется фрагментация; • 6 — неизвестная сеть; • 7 — неизвестный компьютер
8	0	echo-request — такие пакеты используются при проверке связи с компьютером через утилиту ping и запрашивают ответ от хоста
9 и 10	0	Сообщения такого типа рассылают маршрутизаторы
12	1	Неправильный IP-заголовок
12	2	Требуемые опции отсутствуют

4.11.7. Перенаправление

Все правила, которые мы описывали до сих пор, относятся только к доступу к компьютеру. В случае если компьютер используется как выделенный сетевой экран, то тут настройки изменяются, и в основном вы будете работать с forward-записями.

Сетевой экран, который защищает целую сеть, состоит, как минимум, из компьютера с двумя интерфейсами. Один из них (сетевая карта или модем) направлен в Интернет, а другой — в локальную сеть. Связь с Интернетом происходит через этот компьютер, поэтому сетевой экран будет заниматься перенаправлением трафика с одного интерфейса в другой, т. е. с сетевой карты на модем и обратно. Такой компьютер называют шлюзом. Пользователи не подключаются к самому шлюзу, а только используют его как средство переадресации пакетов.

Вы можете установить какие-либо сервисы прямо на сетевой экран, но я не рекомендую этого делать. Лучше, если они будут за пределами этого компьютера. Публичные сервисы лучше ставить со стороны Интернета, а закрытые — должны быть на серверах внутри вашей локальной сети (рис. 4.6).

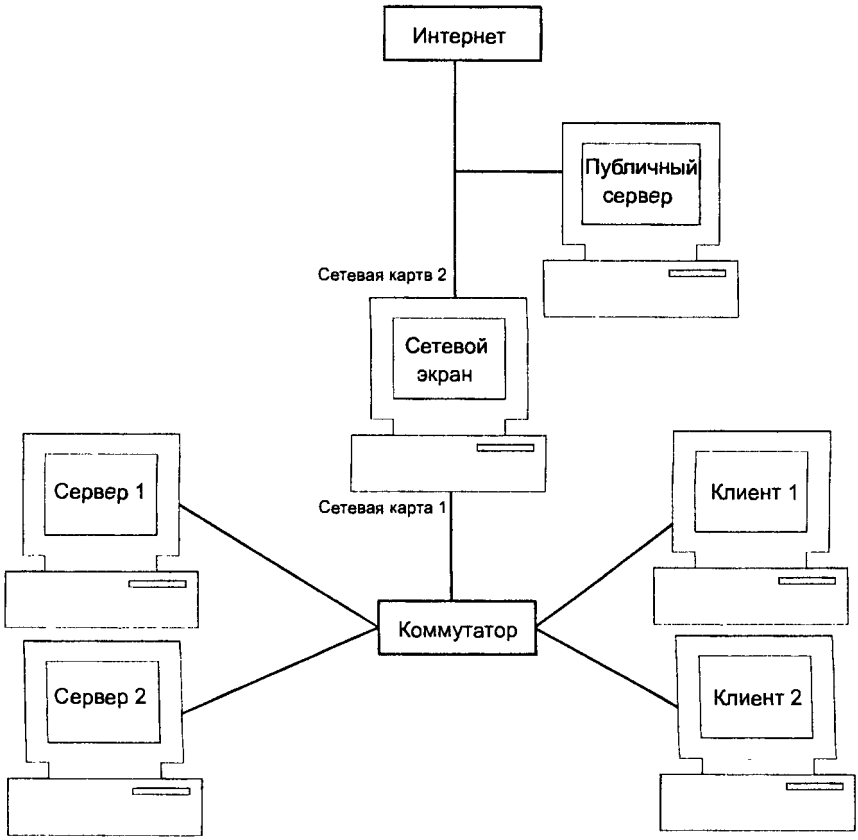

Рис. 4.6. Расположение серверов с публичными и закрытыми сервисами

Подобная схема позволяет для публичных сервисов не прописывать разрешающие записи в сетевом экране, потому что он их и не защищает. Но на таком сервере можно применить локальную политику доступа. Я вообще не рекомендую в своей сети использовать публичные сервисы. Существует множество хостинговых компаний, которые специализируются на предоставлении подобных услуг. Доверьтесь профессионалам, которые снимут часть забот с ваших плеч.

Если сервер с публичным сервисом расположить внутри сети, то в сетевом экране придется прописывать разрешения на доступ к нему для всех пользователей из Интернета. Мы уже рассмотрели подобные права *(см. разд. 4.11.2)*, но это был только пример, а в реальной жизни вседоступные сервисы необходимо вынести за пределы частной сети.

Каждое лишнее разрешение в сетевом экране — это дополнительная дверь внутрь локальной сети. Предположим, что вы сами содержите Web-сервер.

Если в нем или используемых на сервере сценариях будет найдена ошибка, то вся сеть окажется под угрозой. Нельзя допускать, чтобы незначительная погрешность стала причиной больших проблем.

Если вы все же решили самостоятельно обслуживать публичные серверы, то могу порекомендовать организовать сеть, как показано на рис. 4.7. Только в этом случае придется использовать дополнительный сервер для организации второго Firewall.

Рис. 4.7. Двойная защита сети

На данной схеме первый сетевой экран защищает Web-сервер. Его политика будет достаточно мягкой, потому что должна разрешить публичный доступ к некоторым портам сервера, например 80 для путешествий по Web-сайтам. Главное, чтобы фильтры открывали для всеобщего доступа исключительно адрес публичного сервера и только допустимые порты.

Второй сетевой экран защищает локальную сеть, поэтому должен иметь более жесткие правила, запрещающие любые соединения извне. Помимо этого, не должно быть никаких доверительных отношений между публичным сервером и локальной сетью. Если вы разрешите такому серверу свободное подключение к каким-либо портам внутренней сети, то смысл использования такой схемы теряется. Не забывайте, что благодаря ошибке в Web-сервере или сценарии злоумышленник сможет выполнять команды и устанавливать соединения от имени публичного сервера, и хакеру даже не придется взламывать второй Firewall.

Я видел реально построенную схему, как показано на рис. 4.7, но помимо публичных серверов между двумя сетевыми экранами находились несколько компьютеров, которые создавали видимость сети. Это была бутафория из старых машин, которая должна была сбивать хакеров с толку. На этих компьютерах не было ничего ценного, зато были установлены различные средства обнаружения атак (об этом мы поговорим в *гл. 12*), которые сигнализировали администратору, если злоумышленник проникал в систему.

Построение сети из старых компьютеров до какой-то степени запутает хакера. На некоторых машинах можно открыть порты и накидать ненужных файлов, чтобы злоумышленник подольше искал необходимую информацию.

Есть еще один вариант — установить в компьютер три сетевые карты. Одна смотрит в Интернет, ко второй подключена приватная локальная сеть, а к третей — сеть с серверами, на которых работают открытые ресурсы (рис. 4.8). Получается защита в виде вилки. На один интерфейс мы можем перенаправлять любой трафик из Интернета, а другой интерфейс защищаем всеми возможными способами.

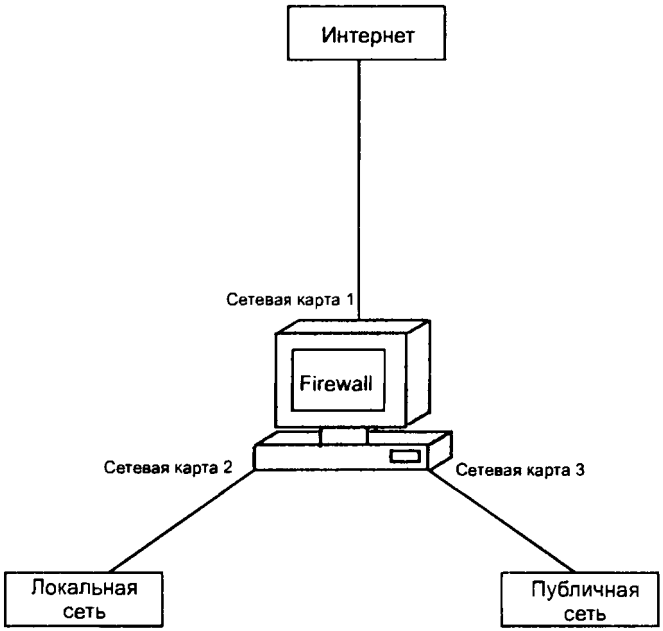

Рис. 4.8. Защита двух сетей одним сетевым экраном

С точки зрения безопасности схема, показанная на рис. 4.7, намного надежнее, потому что локальную сеть защищают сразу два сетевых экрана, и их легче конфигурировать. Второй вариант (рис. 4.8) проще и дешевле, т. к. не требует дополнительного компьютера для Firewall, но менее безопасен. Проникнув на компьютер с сетевым экраном, злоумышленнику потребуется меньше усилий, чтобы взломать приватную сеть.

Вернемся к самой сути перенаправления. На рис. 4.6 изображена сеть на основе витой пары с центральной точкой в виде коммутатора. Давайте проследим связи. Чтобы попасть в Интернет, любой пакет от компьютера проходит через коммутатор, входит через сетевую карту 1 (допустим, что это eth0) на машину с установленным Firewall и выходит через сетевую карту 2 (предположим, что это eth1) в Интернет.

На самом компьютере с Firewall должно быть разрешено перенаправление, которое позволит пакетам проходить из одной сетевой карты в другую. Что-

бы включить эту возможность, запишите в файл **/proc/sys/net/ipv4/ip_forward** число 1 (по умолчанию там может быть 0) или выполните команду:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Для обеспечения переадресации между сетевыми интерфейсами ядро операционной системы должно быть скомпилировано с соответствующей поддержкой, потому что перенаправление происходит именно на этом уровне. Помимо этого, нужно изменить параметр `net.ipv4.ip_forward` в файле **/etc/sysctl.conf** на 1.

Более подробно о предоставлении доступа в Интернет мы поговорим в *гл. 9*. А сейчас затронем только темы, которые касаются настройки безопасности перенаправления трафика.

Сетевой экран может не только проверять пакеты на соответствие определенным фильтрам, но и прятать IP-адреса компьютеров сети. Это происходит следующим образом:

1. Клиент направляет пакет в сеть, и до сетевого экрана он будет идти со своим IP-адресом.
2. Сетевой экран меняет IP-адрес отправителя на свой и направляет пакет дальше от своего имени.

Таким образом, в Интернете все пакеты будут видны, как будто их отправлял Firewall, а не компьютеры внутри сети. Это позволяет скрыть от злоумышленника внутреннюю организацию сети и экономить IP-адреса. Внутри сети вы сами можете раздавать адреса из зарезервированного диапазона (мы их рассматривали в *разд. 4.10.2*), и только сетевой экран будет иметь реальный IP-адрес. При такой адресации из Интернета нельзя будет напрямую подключиться к компьютерам вашей сети. Это значит, что хакеру придется взламывать сначала машину с сетевым экраном и только потом компьютеры сети. Тем самым мы значительно усложняем задачу взломщика. Неопытные хакеры никогда не связываются с сетевыми экранами, потому что для такого проникновения нужны не только широкие знания принципов безопасности, но и большой опыт.

Рассмотрим пример, который разрешает переадресацию на внешний интерфейс из локальной сети:

```
ipchains -A forward -i eth1 -s 192.168.1.0/24 -j MASQ
```

Это общее правило, поэтому я добавляю его в конец цепочки `forward` с помощью ключа `-A`, чтобы оно не перекрыло фильтры, которые относятся к конкретным пользователям, но в то же время взаимосвязаны с этой записью.

Далее идет ссылка на интерфейс `eth1` (сетевая карта, которая смотрит в Интернет). Диапазон адресов соответствует всей сети 192.168.1.x. В качестве

разрешения (ключ -j) используется значение MASQ, что соответствует маскированию адреса, т. е. адрес клиента будет заменен на адрес компьютера, на котором работает Firewall.

Данное разрешение позволяет только передавать пакеты с адресов 192.168.1.x на сетевой интерфейс eth1. Но это еще не значит, что трафик поступит и сможет выйти в Интернет. Чтобы пакеты пользователей были приняты сетевым экраном, должно быть разрешение ACCEPT примерно следующего вида:

```
ipchains -A input -i eth0 -s 192.168.1.0/24 -j ACCEPT
```

В этом фильтре мы открыли доступ на сетевой интерфейс eth0 любым пакетам с адресов 192.168.1.x. Разрешение дано на порты, поэтому адрес отправителя пакета может быть любым из сети 192.168.1.x.

Осталось только разрешить таким же образом пакетам выходить (цепочка output) из eth1, и можно считать, что все компьютеры вашей сети уже получили доступ в Интернет. Только на всех клиентских машинах необходимо указать в качестве шлюза по умолчанию IP-адрес компьютера с сетевым интерфейсом, и все пакеты найдут своего адресата.

Если у вас структура сети напоминает рис. 4.7, то переадресация должна быть включена на обоих сетевых экранах. А вот маскировку адреса лучше всего сделать на Firewall 2. В этом случае второй сетевой экран будет скрывать локальную сеть даже от публичного сервера.

Зачастую в качестве второго сетевого устройства выступает не карта, а модем. В этом случае правило для перенаправления будет выглядеть следующим образом:

```
ipchains -A forward -i ppp0 -s 192.168.1.0/24 -j MASQ
```

Здесь перенаправление идет на интерфейс одного из модемов, имена которых имеют вид pppX.

Чаще всего необходимо, чтобы ваши клиенты имели доступ к ресурсам Интернета, а обратное подсоединение было невозможно. Когда по протоколу TCP запрашивается подключение к удаленному компьютеру, то посылается пакет с установленным битом syn. В простых пакетах (ответы на наши соединения или передача данных) такого бита не должно быть. Таким образом, достаточно запретить TCP-пакеты с этим флагом, и удаленный компьютер не сможет подключиться ни к одному ресурсу нашего компьютера или сети. Это можно реализовать следующим образом:

```
ipchains -I input 1 -i ppp0 -p tcp --syn -j DENY
```

В данной строке вставляется новое правило в цепочку проверки входящих пакетов. Контролируются пакеты TCP, у которых установлен флаг syn (об этом говорит ключ --syn). Если такой пакет получен, то он удаляется.

Для использования маскировки IP-адреса ядро операционной системы должно быть скомпилировано с соответствующей поддержкой, потому что подмена адреса происходит на уровне ядра.

4.11.8. Сохранение фильтра

Если попробовать сейчас перезагрузить систему и просмотреть цепочки сетевого экрана, то все наши изменения исчезнут. Проблема в том, что ОС автоматически их не запоминает, и мы сами должны позаботиться о сохранении правил. Для этого существует утилита ipchain-save. Ее нужно выполнять следующим образом:

```
ipchain-save > файл
```

Когда мы только начинали рассматривать команды ipchains, то я упоминал файл **/etc/sysconfig/ipchains** *(см. разд. 4.11.1)*. Это конфигурационный файл, который загружается при старте системы. Именно в этом файле я рекомендую держать все настройки:

```
ipchain-save > /etc/sysconfig/ipchains
```

Выполняйте сохранение после каждой модификации конфигурации. Если случится внештатная ситуация и сервер придется перезагружать, то вы обязательно забудете восстановить изменения.

Сохранение цепочек можно сделать автоматическим, но я не советую вам на это надеяться. Лучше сделать все вручную, так будет надежнее.

Восстановить цепочки тоже можно из файла. Для этого необходимо выполнить команду:

```
ipchain-restore < файл
```

Это очень удобно. Допустим, что вы хотите протестировать новые правила, но боитесь испортить уже отлаженные цепочки. Сохраните в каком-нибудь файле текущее состояние и изменяйте что угодно и как угодно. В любой момент можно вернуться к исходной точке, выполнив одну команду восстановления.

4.12. iptables

Программа iptables является новой разработкой по управлению фильтрами и обеспечению безопасности, но пока еще не смогла завоевать сердца большинства пользователей. Если вы разобрались с утилитой ipchains, то понять принцип работы с iptables будет не сложнее.

С помощью iptables вы также можете редактировать цепочки правил input, output и forward.

4.12.1. Основные возможности iptables

Сходство между ipchains и iptables прослеживается уже при взгляде на параметры:

- `-A цепочка правило` — добавить правило в конец цепочки. В качестве параметра указывается имя цепочки `INPUT`, `OUTPUT` или `FORWARD`;
- `-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. Здесь можно использовать `INPUT`, `FORWARD` или `PREROUTING`;
- `-o интерфейс` — задает интерфейс, на который направляется пакет. Здесь можно указывать `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;
  - `REGECT` — отправителю будет направлено сообщение об ошибке;
  - `DROP` — удалить пакет;
  - `BLOCK` — блокировать пакеты;
- `-s адрес` — IP-адрес отправителя пакета. Как и в случае с iptables, после адреса можно задать маску в виде `/mask` и знак отрицания "!", что будет соответствовать любым адресам, кроме указанных;
- `-d адрес` — адрес назначения пакета.

Как видите, большинство параметров абсолютно идентичны тем, что мы рассматривали для программы ipchains. Но есть важные и очень мощные отличия. Например, с помощью ключей `-o` и `-i` очень просто указывать, с какого на какой интерфейс направляется пакет. Из-за сходства конфигурирования сервисов ipchains и iptables в практической части мы не будем тратить драгоценное место книги, и кратко рассмотрим создание правил.

В данном обзоре ключей я затронул только основы, но если вы посмотрите файл документации, то увидите еще много вариантов работы с ключом -j, т. е. существуют большие возможности по управлению пакетом, если он соответствует правилам.

Настройка цепочек iptables не сильно отличается от ipchains. Начать формирование цепочки нужно с очистки всего содержимого. Двигаться необходимо от полного запрета и разрешать только то, что не нанесет вреда серверу. Сервисы, которые могут оказаться опасными, должны быть доступны только тем, кому это необходимо, или тем, кому вы доверяете. Хотя, в нашем деле полагаться ни на кого нельзя. Ваш друг может, не желая того, раскрыть секретную информацию злоумышленнику, или данные могут быть просто украдены, и тогда доверчивость сыграет с вами злую шутку.

Для сохранения изменений в iptables также надо выполнить специализированную команду:

```
service iptables save
```

4.12.2. Переадресация

Для разрешения переадресации с помощью iptables нужно выполнить следующую команду:

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

В данной строке позволяется переадресация на интерфейс ppp0. С помощью параметра -j мы требуем прятать IP-адрес отправителя, т. е. включаем маскарадинг.

Если вы используете трансляцию сетевых адресов (NAT, Network Address Translation), то 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
```

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

4.12.3. Примеры конфигурирования iptables

Я не буду подробно останавливаться на описании различных запретов, потому что мы о них говорили при рассмотрении программы ipchains. Мы очень коротко рассмотрим создание различных правил.

Запрет любых обращений будет выглядеть следующим образом:

```
iptables -P INPUT DROP
```

Теперь все входящие пакеты будут удаляться. Как и в случае с программой ipchains, именно с этой команды нужно начинать конфигурирование iptables. Обратите внимание, что в правиле используется ключ `-P`, позволяющий задать значение по умолчанию для данной цепочки. Если фильтр добавить с помощью ключа `-A`, то вы можете запретить абсолютно любые подключения.

Некоторые специалисты по безопасности рекомендуют журналировать обращения, добавив в сетевой экран фильтр:

```
iptables -A INPUT -j LOG
```

Я бы не рекомендовал это делать. У публичных серверов за день происходит несколько сотен, а то и тысяч сканирований портов. Если обращать внимание на каждую такую попытку, вам придется устанавливать на сервер слишком большие жесткие диски для хранения журналов. А ведь если диск будет заполнен, то система выйдет из строя. Таким образом, хакер может просто направить бесконечные обращения на запрещенный порт и через некоторое время добиться удачно завершенной DoS-атаки.

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

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

Как видите, создание фильтра с помощью iptables не сильно отличается от аналогичной процедуры в ipchains.

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
```

Теперь запретим исходящие пакеты с 21 порта. Для этого используем команду:

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

Очень мощной особенностью iptables является возможность проверки содержимого пакетов. Это очень удобно, например, для фильтрации Web-запросов.

Можно разрешить доступ к 80 порту, но контролировать, чтобы пакеты содержали только допустимые параметры. К безопасности Web-сервера мы вернемся в *гл. 7* и познакомимся с разными способами защиты. А сейчас рассмотрим простой, но универсальный.

Допустим, что мы хотим разрешить доступ к FTP-серверу, но при этом быть уверенными, что пользователь не сможет обратиться к файлам **/etc/passwd** и **/etc/shadow**. Для этого можно запретить пакеты, в которых есть этот текст. Если хакер попытается послать запрос, содержащий ссылки на эти файлы, то такой пакет будет отклонен. Следующие команды запрещают доступ к этим файлам по протоколам FTP и WWW:

```
iptables -A INPUT -m string --string "/etc/passwd" \
-s 0/0 -d localhost -p tcp --dport 21 -j DROP
iptables -A INPUT -m string --string "/etc/shadow" \
-s 0/0 -d localhost -p tcp --dport 21 -j DROP
iptables -A INPUT -m string --string "/etc/passwd" \
-s 0/0 -d localhost -p tcp --dport 80 -j DROP
iptables -A INPUT -m string --string "/etc/shadow" \
-s 0/0 -d localhost -p tcp --dport 80 -j DROP
```

Надо еще учесть аспект защиты информации. Допустим, что у вас есть сервер, который принимает закодированный трафик с помощью stunnel (Security Tunnel, безопасный туннель, создающий шифрованный канал между двумя машинами, будем рассматривать в *разд. 5.2*), расшифровывает и передает его на другую машину. В этом случае во входящих пакетах сетевой экран не может найти такие строки. А вот исходящие пакеты идут уже декодированными и содержат открытый текст команд. В такой конфигурации необходимо контролировать оба потока.

Даже если у вас stunnel передает расшифрованный трафик на другой порт внутри одного компьютера, можно включить контроль любых пакетов на всех интерфейсах, чтобы они проверялись после расшифровки.

4.13. Замечания по работе Firewall

Сетевой экран может как защитить вашу сеть или компьютер от вторжения, так и сделать ее уязвимой. Только внимательное конфигурирование и жестокие запреты сделают вашу систему надежной.

Но даже если вы очень вдумчиво все настроили, нет гарантии, что сервер окажется в безопасности. Абсолютная неуязвимость сетевого экрана — это миф. И в данном случае проблема заключается не только в программах iptables или ipchains. Сама технология сетевого экрана не гарантирует полной

безопасности. На 100 % ее никто не может обеспечить, иначе я не писал бы эту книгу.

В данном разделе нам предстоит познакомиться с проблемами, с которыми вы можете встретиться во время использования сетевого экрана. Вы должны четко себе их представлять, чтобы знать, откуда может идти угроза.

4.13.1. Внимательное конфигурирование

Как я уже сказал, только предельная внимательность может обеспечить относительно спокойный сон администратора и специалиста по безопасности. Давайте разберем наиболее типичные промахи администраторов, это поможет избежать появления подобных ошибок в вашей практике.

Как вы помните, теперь у нас по три записи для цепочек input и output. А что, если вам уже не нужен больше FTP-доступ, и вы хотите его убрать. Отключив FTP-сервер, не забудьте удалить разрешающие правила из chains-цепочек.

В моей практике был случай, когда знакомый администратор не убрал эти записи. Через какое-то время доступ снова был включен, но под разрешенным IP-адресом работал уже другой пользователь. Для опытного администратора это вполне знакомая ситуация, и сервер попал под угрозу. Хорошо, что IP достался человеку, который и не собирался делать ничего разрушительного.

Очень тяжело, если IP-адреса распределяются динамически и могут регулярно меняться. Если в вашей сети используется сервер DHCP (Dynamic Host Configuration Protocol, протокол динамической конфигурации хоста), то нужно позаботиться о том, чтобы компьютеры, которым необходим особый доступ и правила, имели жестко закрепленный адрес (например, основного шлюза). Это предотвратит случайное попадание IP-адреса в недобросовестные руки и потерю привилегий теми, кто в них нуждается.

Представьте себе, что если в нашем примере, который мы рассматривали при создании chains-цепочек *(см. разд. 4.11.2)*, IP-адрес 192.168.8.10 случайно окажется у другого компьютера? Возникнут очень большие проблемы, потому что реальный владелец адреса потеряет доступ, а новый хозяин его получит.

Чтобы четко контролировать IP-адреса, желательно использовать DHCP-сервер и жестко фиксировать адреса тем, кто нуждается в привилегированном доступе и имеет фильтры в цепочках.

При формировании правил будьте очень внимательны. Некоторые сервисы (такие как FTP) могут требовать для своей работы более одного порта. Если вы не откроете/закроете все порты, то можете не получить необходимого результата.

При настройке сетевого экрана из графической оболочки будьте особенно осторожны. При запрете всего XWindow может зависнуть, если не найдет сетевого соединения с ядром Linux.

Я конфигурирую свой сервер через удаленное соединение по протоколу SSH. Тут тоже нужно быть аккуратным, потому что одно неверное действие может оборвать подключение и SSH-клиент теряет связь. После этого приходится идти в серверную комнату и настраивать сетевой экран прямо там.

Тестируйте все используемые соединения после каждого изменения конфигурации сетевого экрана. Внеся несколько модификаций очень тяжело найти ошибку.

Для поиска конфликтных цепочек я сохраняю конфигурацию во временный файл (любой, кроме системного **/etc/sysconfig/ipchains**) и распечатываю ее. На бумаге намного проще, чем на экране монитора, видеть всю картину в целом. Обращайте внимание на правильность указания параметров (адрес и порт) источника и получателя пакета. Очень часто администраторы путают, где и что прописывать.

Мысленно пройдите по каждой записи, анализируя, какие пакеты пропускаются, а какие нет. Удобней начинать обследование с цепочки input (когда пакет входит в систему). Затем проверяйте перенаправление и, наконец, выход, т. е. цепочку output. Таким образом, нужно проследить полный цикл прохождения пакета. Помните, что после первой найденной записи, соответствующей параметрам пакета, дальнейшие проверки не производятся.

При контроле записей, относящихся к TCP, помните, что этот протокол устанавливает соединение, а значит, необходимо, чтобы пакеты проходили в обе стороны. Протокол UDP не требует подключения, и пакеты можно пропускать в одну сторону — input или output. Но бывают исключения, некоторым программам нужен двусторонний обмен даже по протоколу UDP.

Если какая-либо программа не работает, то убедитесь, что существуют правила для всех необходимых портов. Некоторые протоколы требуют доступ к двум и более сетевым портам. После этого проверьте, чтобы запись с разрешением шла раньше фильтра запрещения.

Никогда не открывайте доступ к определенному порту на всех компьютерах. Например, если просто добавить фильтр разрешения для входящих на 80 порт пакетов, то в результате этого канал будет открыт на всех компьютерах сети. Но далеко не всем он необходим. При формировании правила указывайте не только порт, но и конкретные IP-адреса, а не целые сети.

Регулярно делайте резервную копию цепочек. Для этого можно сохранять их содержимое в отдельном файле с помощью команды `ipchain-save` или копировать файл **/etc/sysconfig/ipchains** (желательно, на другой компьютер). Тогда

в случае возникновения проблем можно быстро восстановить хорошие цепочки, а тестирование сетевого экрана с новыми параметрами перенести на внерабочее время.

4.13.2. 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 была бы скучной и неинтересной.

Как Firewall 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 безопасность, и нет такого правила, которое нельзя обойти.

На большинстве сетевых экранов очень легко реализовать атаку DoS. Когда мы рассматривали технологию этой атаки *(см. разд. 1.1.6)*, то говорили о том, что она легко организуется в двух случаях:

1. Мощность вашего канала больше, чем у противника.
2. 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 sun-пакетов больших ресурсов и мощного канала не надо. Хакер без проблем может забросать разрешенный порт сервера sun-пакетами, в которых адрес отправителя подставляется случайным образом. Процессор атакуемой машины может не справиться с большим потоком запросов, которые надо сверять с фильтрами, и выстроится очередь, которая не позволит обрабатывать подключения добропорядочных пользователей.

Самое страшное, если сетевой экран настроен на отправку сообщений с ошибками. В этом случае нагрузка на процессор увеличивается за счет создания и посылки пакетов на несуществующие или не принадлежащие хакеру адреса.

Если клиент посылает слишком много данных, которые не могут быть помещены в один пакет, то информация разбивается на несколько блоков. Этот процесс называется фрагментацией пакетов. Большинство сетевых экранов анализируют только первые блоки в сессии, а все остальные считаются пра-

вильными. Логика такого контроля понятна, если первый пакет верен, то зачем проверять их все и тратить на это драгоценные ресурсы сервера? В противном случае от остальных не будет толка, потому что соединение не установлено и нарушена целостность информации.

Чтобы сетевой экран пропустил данные хакера, пакеты могут быть специальным образом фрагментированы. От подобной атаки можно защититься, только если Firewall осуществляет автоматическую сборку фрагментированных пакетов и просматривает их в собранном виде. В большинстве сетевых экранов такая возможность отсутствует.

Сетевой экран очень часто становится объектом атаки, и не факт, что попытка не окажется успешной. Если злоумышленнику удастся захватить Firewall, то сеть станет открытой, как на ладони. В этом случае вас смогут спасти от тотального разгрома только персональные сетевые экраны на каждом компьютере. На практике политика безопасности на персональном компьютере не такая жесткая, но может быть вполне достаточной для предотвращения дальнейшего проникновения хакера в сеть.

Атака на сетевой экран не зависит от его реализации. Ошибки бывают как в ОС Linux, так и в маршрутизирующих устройствах с возможностями фильтрации.

Основная задача, которую решает сетевой экран, — запрет доступа к заведомо закрытым ресурсам. Но существуют открытые ресурсы. Например, если необходимо, чтобы Web-сервер был доступен пользователям Интернета, то сетевой экран не сможет защитить от взлома через ошибки в сценариях на Web-сервере.

Максимальная безопасность приносит некоторые неудобства. Так, я уже говорил, что лучше всего запретить любые попытки подключения извне. Соединение может быть установлено только по инициативе клиента вашей сети, но не удаленного компьютера. В этом случае хакер останется за бортом, но и у пользователей сети могут возникнуть проблемы, например, при попытке подсоединения к FTP-серверу в активном режиме. Мы уже знаем, что этот сервис работает на двух портах: ftp и ftp-data (ftpd). Пользователь подключается к серверному порту ftp, а когда вы запрашиваете получение файла, сервер сам инициирует соединение с клиентом, а этого сетевой экран не разрешит. Для FTP-сервиса решили эту проблему, добавив возможность работы в пассивном режиме, но в других программах (например, в чатах) вопрос остается открытым.

Хакер может установить соединение с защищенной сетью через туннель на открытом порту и с дозволенным адресом внутри сети. От этого уже никуда не денешься, потому что хоть что-то, но должно быть разрешено.

В крупных компаниях в одной сети может быть несколько серверов. Я только в одной фирме и в кино видел, как администраторы для управления каждым из них работают за несколькими мониторами и клавиатурами одновременно. В реальной жизни такие специалисты слишком ленивы, да и однообразный труд утомляет, поэтому они сидят только за одним компьютером, а для подключения к серверу используют удаленное соединение.

Но на этом лень администраторов не заканчивается. Чтобы не приезжать на работу во внеурочное время в случае экстренной ситуации, им требуется доступ к консоли сервера прямо из дома. А вот это уже может стать сильной угрозой. Благо, если программа, через которую происходит управление, поддерживает шифрование (например, SSH), а если это простой telnet-клиент? Злоумышленник сможет подсмотреть параметры аутентификации с помощью утилиты сниффинга и получить такой же административный доступ к серверу.

4.13.3. 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.

Вы должны тщательно маскировать, что именно и кому разрешено на сервере. Чем меньше знает хакер, тем лучше. Помимо этого, должны пресекаться любые разведывательные действия, например, использование сканеров портов, трассировка сети и др.

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 хакеры стали использовать для диагностики сети, чтобы узнать маршрут, по которому проходит пакет. Как это работает? В 99 % случаев каждый запрос идет до адресата одним и тем же путем. На отправленный со значением TTL, равным 1, пакет 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 Firewall. Пакет с реальным значением TTL дойдет до адресата, а если команда `traceroute` выдала ошибку, то это значит, что на пути следования пакета есть Firewall, который запрещает трассировку.

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

```
traceroute -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.

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

```
traceroute 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]
...
...
17   638 ms   839 ms   479 ms  216.140.3.38

18     *        *        *     Request timed out.
```

Если сетевой экран допускает ICMP-пакеты, то сканирование можно провести с помощью `traceroute`. Возможно, появится сообщение об ошибке. В данном случае 18 строка сообщает о превышении времени ожидания ответа. Это значит, что пакет отправлен, но сервер отбросил запрос, а значит, пакет с TTL, равным 18, будет уничтожен.

Для сканирования сети за пределами Firewall достаточно выполнить команду соединения с компьютерами внутри сети с TTL, равным 19. Во время трассировки мы увидим первые 17 ответов, 18 пропадет, а 19 пройдет дальше в сеть, потому что на сетевом экране такой пакет появится с TTL=2 и не будет удален, а вот в локальной сети первый же маршрутизатор вернет ошибку.

Но в реальности ICMP-пакеты запрещены, поэтому такой метод редко приносит злоумышленнику пользу.

С другой стороны, если мы увидели полный путь к компьютеру назначения, это еще не значит, что сетевого экрана нет. Он может просто не запрещает ICMP-трафик.

Внутреннюю сеть можно просканировать и через DNS-сервер, если он находится внутри нее и доступен для всеобщего использования.

4.13.4. Дополнительная защита

Помимо фильтров на основе определенных администратором правил в сетевом экране может быть реализовано несколько дополнительных защитных механизмов, которые работают вне зависимости от вашей конфигурации или могут включаться специальными опциями. Рассмотрим наиболее интересные аспекты такой защиты.

Одним из популярных методов обхода Firewall и проведения атаки является фальсификация IP-адреса отправителя. Например, у хакера может быть адрес 100.1.1.1, но с него запрещено подключаться к сервису FTP. Чтобы получить доступ, хакер может послать пакеты, в которых в качестве отправителя указан, например, 100.2.2.2.

Но это еще не все. Просто воспользовавшись чужим именем, ничего хорошего не получится. *Хакер не увидит отклика сервера* (рис. 4.9).

Чтобы хакер смог получить ответ на свой запрос, в IP-пакет должна быть добавлена специальная информация, по которой сервер найдет реальный адрес хакера 100.1.1.1.

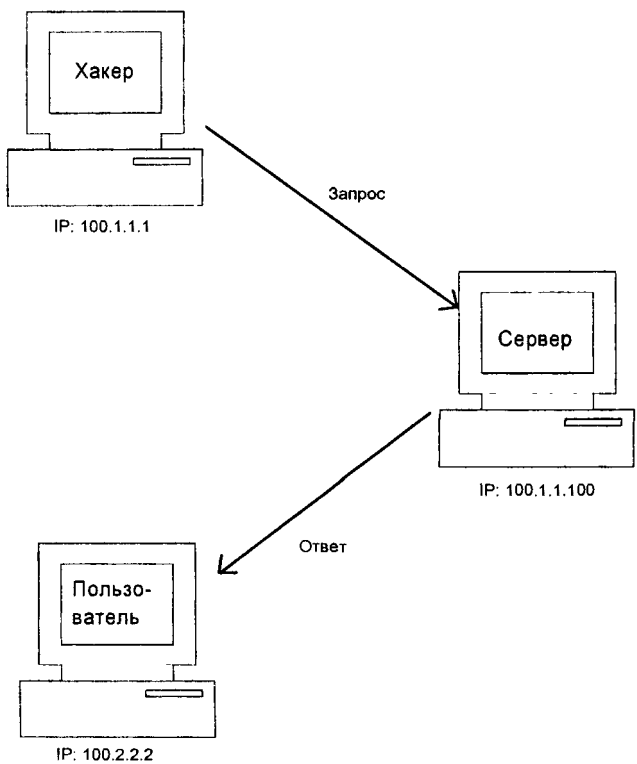

Рис. 4.9. Подмена IP-адреса

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

4.14. Запрет и разрешение хостов

Работа с ipchains или iptables *(см. разд. 4.11 и 4.12)* 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 (например, временная защита) 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 для компьютера нет ни одной записи в файлах, то доступ по умолчанию разрешен.
2. Если соответствие обнаружено в файле hosts.allow, то доступ разрешен и файл hosts.deny не проверяется.
3. Если в файле hosts.deny найдена запись, то доступ запрещен.

Удобство использования этих файлов заключается в том, что в них нужно указывать сервисы, требующие ограничения доступа. Это делается в виде строк следующего вида:

```
сервис: хост
```

Строка состоит из двух параметров, разделенных двоеточием. Первым указывается имя сервиса (или список, разделенный запятыми), доступ к которому нужно ограничить. Второй — это адреса (для файла **/etc/hosts.allow** разрешенные, а для **/etc/hosts.deny** — запрещенные), разделенные запятыми. В качестве параметров можно использовать ключевое слово `ALL`, которое соответствует любому адресу или сервису.

Рассмотрим пример конфигурирования файла. Для начала закроем любой доступ. Для этого в файле **/etc/hosts.deny** нужно прописать запрет для всех пользователей на любые сервисы. Для этого добавляем строку `ALL: ALL`. В результате ваш файл будет выглядеть следующим образом:

```
#
# hosts.deny    This file describes the names of the hosts which are
#               *not* allowed to use the local INET services, as decided
#               by the '/usr/sbin/tcpd' server.
#
```

```
# The portmap line is redundant, but it is left to remind you that
# the new secure portmap uses hosts.deny and hosts.allow.  In particular
# you should know that NFS uses portmap!

ALL: ALL
```

Теперь санкционируем только следующий доступ:

- компьютеру с адресом 192.168.1.1 разрешено подключение ко всем сервисам;
- с ftpd-сервисом могут работать только компьютеры с адресами 192.168.1.2 и 192.168.1.3.

```
#
# hosts.allow  This file describes the names of the hosts
#           which are allowed to use the local INET services,
#           as decided by the '/usr/sbin/tcpd' server.
#

ALL: 192.168.1.1
ftpd: 192.168.1.2, 192.168.1.3
```

Если вам нужно целой сети позволить доступ к какому-либо сервису, то можно указать неполный адрес:

```
ftpd: 192.168.1.
```

В данной строке разрешен доступ к ftpd-сервису всем компьютерам сети 192.168.1.x (последнее число адреса не указано, значит, оно может быть любым).

Как видите, использовать файлы **/etc/hosts.allow** и **/etc/hosts.deny** намного проще, потому что не требуется прописывать правила для входящих и исходящих пакетов. Но возможности этих файлов слишком ограничены и намного меньше, чем у любого сетевого экрана.

Я рекомендую использовать файлы **/etc/hosts.allow** и **/etc/hosts.deny** для решения временных проблем безопасности. Если найдена уязвимость в каком-либо сервисе, 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 временные правила я не рекомендую устанавливать в сетевом экране.

4.15. Советы по конфигурированию Firewall

Конфигурирование сетевого экрана достаточно индивидуально и зависит от конкретных задач, решаемых сервером. Но все же дам некоторые рекомендации, 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 будет проходить с большими сложностями;
- если есть возможность, необходимо запретить все типы ICMP-сообщений, особенно ping. Мы еще не раз будем говорить об опасности сканирования сети с помощью ICMP-пакетов;
- запретить доступ к 111 порту. 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
  Программ вер проток порт
  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 выдать достаточно много информации, поэтому 111 порт необходимо закрыть;
- для облегчения управления доступом к портам разделите открытые ресурсы на две категории:
  - для всеобщего просмотра, в том числе и пользователями Интернета;
  - только для использования внутри сети. Например, такие сервисы, как ftp и telnet, несут в себе опасность, потому что позволяют закачивать файлы на сервер и выполнять на нем команды. Если пользователям Интернета нет необходимости в этих службах, то следует их явно запретить для внешних подключений.

4.16. Повышение привилегий

В заключение рассмотрения темы безопасности необходимо подробно познакомиться с командой sudo, которая позволяет выполнять программы от имени другого пользователя.

Мы уже говорили в *разд. 2.7* о том, что нельзя работать в системе под учетной записью администратора. Это опасно по следующим причинам:

- программы, запущенные вами таким образом, работают с правами администратора. При наличии уязвимости хакер сможет ею воспользоваться для получения полных прав;
- ошибки ввода какой-либо команды могут нарушить работу всей системы. А оплошности бывают часто, потому что в ОС Linux поддерживаются достаточно мощные возможности по использованию регулярных выражений.

Если у вас в системе нет пользователя, не обладающего правами администратора, то добавьте его сейчас. Теперь войдите в систему под его учетной записью и попробуйте просмотреть файл **/etc/shadow**, например, с помощью следующей команды:

```
cat /etc/shadow
```

В ответ на это вы должны получить сообщение о недостатке прав доступа. Теперь выполните ту же команду через sudo:

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

Вы увидите сообщение о том, что ваша учетная запись отсутствует в файле **/etc/sudoers**, в котором прописываются разрешения на использование команды sudo. Пример содержимого этого конфигурационного файла приведен в листинге 4.2.

Листинг 4.2. Содержимое конфигурационного файла /etc/sudoers

```
# sudoers file.
# Файл sudoers
# This file MUST be edited with the 'visudo' command as root.
# Этот файл должен редактироваться с помощью команды 'visudo'
# от имени root
# See the sudoers man page for the details on how to write a sudoers
file.
# Смотрите страницу sudoers руководства, где имеется подробная информация
по использованию sudoers-файла.

# Host alias specification

# User alias specification
```

```
# Cmnd alias specification

# Defaults specification

# 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, получит доступ к выполнению опасных команд;
- компьютер — имя машины, на которой можно выполнять команду от лица администратора;
- команды, которые разрешено выполнять указанному пользователю (перечисляются 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. Чаще в этом участвуют множество людей. Один отвечает за саму систему, другой занимается поддержкой Web-сервера, третий настраивает базу данных MySQL. Давать трем администраторам полные права не имеет смысла, необходимо разрешить каждому выполнять только те команды, которые необходимы для реализации поставленных задач. Таким образом, нужно четко прописывать права для конкретного пользователя.

```
robert    ALL=/bin/cat /etc/shadow
```

Обратите внимание, что я указываю полный путь к программе `cat` (напоминаю, его можно узнать с помощью команды `which`), это необходимо, иначе вместо результата, полученного по разрешенной команде, пользователь robert увидит сообщение об ошибке в конфигурации.

Допустим, вы хотите расширить права пользователя и позволить ему не только просматривать файл паролей, но и монтировать CD-ROM-диск. Для этого изменяем строку, добавляя разрешение на выполнение команды `mount`:

```
robert    ALL=/bin/cat /etc/shadow, /bin/mount
```

Обратите внимание, что в случае с доступом к файлу **/etc/shadow** мы дали добро только на его просмотр, явно указав утилиту cat с параметром в виде пути к файлу с паролями. Это логично, ведь нет смысла изменять его, когда для этого существует команда `passwd`. Можно задать просто разрешение на выполнение команды `cat`:

```
robert    ALL=/bin/cat, /bin/mount
```

Но в этом случае хакер сможет от имени root просматривать любые файлы в системе и даже те, которые не должны быть ему видны.

Для команды `mount` мы не указываем ничего, кроме самой программы. Таким образом, пользователь сам может варьировать ее параметры. Если явно указать в качестве аргумента CD-ROM, то пользователь сможет монтировать именно это устройство:

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

В рассмотренных примерах вместо имени компьютера я всегда применял ключевое слово `ALL`, что соответствует любой машине. Никогда не используйте такое значение параметра в своей реальной системе. Всегда указывайте конкретный компьютер, к которому относится данная запись. Чаще всего это локальный сервер.

С помощью утилиты sudo можно выполнять команды от лица различных пользователей. Для этого используется ключ `-u`. Например, следующая команда пытается просмотреть файл паролей от имени пользователя flenov:

```
sudo -u flenov cat /etc/shadow
```

Если пользователь не указан, то программа sudo по умолчанию запрашивает пароль root. Это не очень удобно, т. к. придется отдавать пароль администратора учетной записи robert. В этом случае теряется смысл в построении такой сложной системы безопасности, ведь зная пароль root, пользователь сможет зарегистрироваться в системе как администратор и сделать все, что угодно.

Никогда не передавайте 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 очень опасно, если вы не перечисляете необходимые директивы, а указываете ключевое слово ALL:

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

Если хакер получит доступ к учетной записи robert, то сможет с помощью утилиты sudo выполнять в системе любые команды. Если вы перечисляете возможные директивы, то серьезность 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. Никогда не делайте этого. Если текстовый редактор запустится для правки даже безобидного файла, хакер получит слишком большие возможности:

- выполнять системные команды. Так как редактор открывается с правами root, команды также будут выполняться от имени этого пользователя, а значит, хакер получит в свое распоряжение всю систему;
- открыть любой другой файл, пользуясь правами администратора.

Я никогда не делегирую возможность корректировки конфигурационных файлов с помощью редакторов. Если без этого не обойтись, то никогда не использую в этом случае права root. Конфигурационному файлу назначается другой владелец, и пользователь для исправлений будет запускать программу sudo только от его имени, а это значит, что редактор будет работать не с правами root.

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 троянские программы, 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.

Напоминаю, что вы должны быть очень внимательны при работе с самой программой sudo, потому что для нее установлен SUID-бит, а значит, она будет выполняться с правами владельца, т. е. администратора системы. Версии sudo, начиная от 1.5.7 и до 1.6.5.p2, содержат ошибку выделения памяти. Хакер может воспользоваться этим для выполнения атаки переполнения стека. Проверьте вашу версию с помощью вызова команды sudo с ключом -v. Если вы являетесь администратором, то увидите на экране подробную информацию о программе, как в листинге 4.3.

Листинг 4.3. Информация о программе sudo

```
Sudo version 1.6.5p2

Authentication methods: 'pam'
Syslog facility if syslog is being used for logging: authpriv
Syslog priority to use when user authenticates successfully: notice
Syslog priority to use when user authenticates unsuccessfully: alert
Ignore '.' in $PATH
Send mail if the user is not in sudoers
Use a separate timestamp for each user/tty combo
Lecture user the first time they run sudo
Require users to authenticate by default
Root may run sudo
Allow some information gathering to give useful error messages
```

```
Visudo will honor the EDITOR environment variable
Set the LOGNAME and USER environment variables
Length at which to wrap log file lines (0 for no wrap): 80
Authentication timestamp timeout: 5 minutes
Password prompt timeout: 5 minutes
Number of tries to enter a password: 3
Umask to use or 0777 to use user's: 022
Path to mail program: /usr/sbin/sendmail
Flags for mail program: -t
Address to send mail to: root
Subject line for mail messages: *** SECURITY information for %h ***
Incorrect password message: Sorry, try again.
Path to authentication timestamp dir: /var/run/sudo
Default password prompt: Password:
Default user to run commands as: root
Path to the editor for use by visudo: /bin/vi
Environment variables to check for sanity:
        LANGUAGE
        LANG
        LC_*
Environment variables to remove:
        BASH_ENV
        ENV
        TERMCAP
        ...
        ...
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 / 0xffffff00
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 полностью приводить результат выполнения команды, но основную информацию можно увидеть. Вначале нам сообщается версия программы

sudo, в данном 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 задается время сохранения пароля в кэше. В данном 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 нелегкой работе. Нам предстоит познакомиться с множеством команд Linux, научиться их использовать и узнать немало полезного о системе. Конечно же, весь рассказ будет сопровождаться интересными фактами и различными примерами.

Но разбором команд мы не ограничимся, потому что иначе книга превратится в перевод руководства по Linux. Чтобы этого не случилось, я постарался подобрать готовые решения задач администратора, которые вам пригодятся в повседневной жизни. Надеюсь, что материалы этой главы помогут вам найти ответы на многие вопросы и избавиться хотя бы от части проблем.

Некоторые специалисты считают, что администрирование — это достаточно простой процесс, только знай команды и выполняй их в нужное время и в нужном месте. Это действительно так, если иметь в виду управление в чистом виде. Но если говорить об обеспечении безопасности системы, то все намного серьезнее, чем может присниться в самом кошмарном сне. И прочитав эту главу, вы с этим согласитесь.

На просторах Интернета идет самая настоящая война между защитой и взломом, и побеждает тот, кто быстрее реагирует на действия противника и меньше спит.

5.1. Полезные команды

Познакомимся с некоторыми программами и командами, которые позволят вам упростить администрирование и сделать его эффективнее. Начнем с команд, необходимых для последующего понимания материала.

5.1.1. netconf

Эта команда запускает конфигуратор сети (рис. 5.1). Программа netconf имеет удобный графический интерфейс и позволяет настраивать сетевые параметры, не задумываясь о конфигурационных файлах.

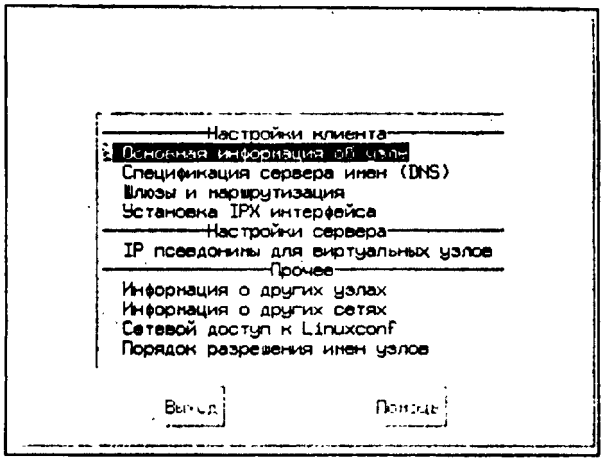

Рис. 5.1. Окно программы netconf

5.1.2. ping

Одна из часто используемых администраторами команд — `ping`. Она посылает ICMP-пакеты в виде эхо-запросов на указанную систему для определения пропускной способности сети.

Например, выполните команду `ping 195.18.1.41` и в ответ вы получите следующую информацию:

```
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
```

Первая строка информационная, в ней отображается IP-адрес компьютера, с которым будет происходить обмен сообщениями (если вы указали символьное имя хоста, то простой командой `ping` можно узнать его IP-адрес). В конце строки находится размер пакетов данных, которые будут отправляться.

Следом на экране начнут появляться строки типа:

```
64 bytes from 195.18.1.41: icmp_seq=1 ttl=64 time=0.102 ms
```

Отсюда мы узнаем, что было получено 64 байта с адреса 195.18.1.41. После двоеточия отображаются следующие параметры:

- `icmp_seq` — номер пакета. Это значение должно последовательно увеличиваться на 1. 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 быть из-за неустойчивой работы оборудования, плохого кабельного соединения или 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 числом. По умолчанию в большинстве систем `ttl` равно 64, но значение может быть изменено. Каждый маршрутизатор при передаче пакета уменьшает это число. 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 о 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. Все остальные должны идти равномерно.

Если ответ на какой-нибудь пакет 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, и после этого можно прерывать сеанс нажатием клавиш <Ctrl>+<C>. В результате вы увидите небольшую статистику обмена сообщениями: количество отправленных, полученных и потерянных пакетов, а также минимальное, среднее и максимальное время обмена пакетами.

Рассмотрим основные параметры, которые можно использовать в команде `ping`:

- `-c n` — завершить работу 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 (скорость достигается за счет того, что все пакеты посылаются в сеть без ожидания ответа). Например, вам нужно быстро отправить 50 запросов, тогда команда будет выглядеть следующим

образом: `ping -f -c 50 195.10.14.18`. Этот ключ 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` — устанавливает размер пакета. Например, вы хотите отправить пакет размером в 1000 байт. В этом случае команда будет выглядеть `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 этому вопросу можно получить в справочной системе, выполнив команду `man ping`.

Внимание!

Не каждый сервер может отзываться 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.

5.1.3. netstat

Эта команда показывает текущие подключения к серверу. 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
```

Вся информация разложена по колонкам. Давайте рассмотрим каждую из них:

- `Proto` — базовый протокол, который использовался для соединения. Чаще всего здесь можно видеть `unix` или `tcp`;
- `Recv-Q` — количество байтов, не скопированных пользовательской программой;
- `Send-Q` — количество байтов, не принятых удаленным компьютером;
- `Local Address` — локальный адрес формата `компьютер:порт`. В качестве порта может использоваться как символьное имя, так и число. В приведенном выше 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 много возможных параметров, и полное их описание можно увидеть в файле справки, набрав команду `man netstat`.

В случае непредвиденной ситуации и подозрение на проникновение в систему извне вы с помощью этой команды легко сможете определить, через какой сервис произошло вторжение, и что хакер в данный момент может использовать. Например, если злоумышленник пробрался через FTP, то он, скорей всего, работает с файлами и может закачивать свои программы для дальнейшего взлома или удалять системные файлы (это зависит от прав доступа).

5.1.4. lsof

Позволяет просмотреть открытые файлы. Команда достаточно мощная и имеет множество параметров. Одна из интересных особенностей — возможность просмотра открытых портов, если указать ключ `-i`:

```
lsof -i
```

Я советую сейчас остановиться и изучить документацию по этой команде, предоставляемую утилитой man.

5.1.5. Telnet

Мощность Linux и его текстовой консоли заключается в том, что вы можете выполнять все действия не только сидя непосредственно за терминалом, но и удаленно. Нужно только подключиться к компьютеру на порт Telnet-сервера с помощью Telnet-клиента, и можно считать, что вы в системе.

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 обладает незначительными возможностями. 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 за сервером. Но этот метод отнимает слишком много трафика и очень неудобен на каналах медленной связи.

Командная строка Linux по сравнению с графическим режимом любой ОС вообще не расходует трафик и может приемлемо работать даже по самым медленным каналам, например, по соединениям GPRS сотового телефона или домашнего модема, где скорость достаточно мала.

Как вы уже поняли, программное обеспечение Telnet состоит из серверной части и клиентской. При запуске Telnet-сервера открывается 23 порт, к кото-

рому можно подключиться с клиентского компьютера и выполнять любые команды, которые позволяет сервер (в данном случае Telnet-сервер).

Но это не все, с помощью Telnet-клиента можно подсоединяться и к другим серверам. Например, можно подключиться на порт 25 и прямо из командной строки отправить E-mail-сообщение, исполняя команды SMTP-сервера.

Если у вас есть установленный FTP-сервер, то уже сейчас вы можете выполнить команду:

```
telnet localhost 21
```

В данном случае первый параметр — 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 21 порту, поэтому я указал это число.

Я рекомендую применять Telnet-клиент только для отладки различных сервисов, но не для управления. Поэтому на всех машинах отключайте Telnet. Он небезопасен, потому что передает данные в открытом виде, а все попытки сделать Telnet защищенным заканчивались неудачно и не приводили к желаемому результату. Один из возможных вариантов использования — через канал шифрования OpenSSL (от Secure Socket Layer, протокол защищенных сокетов). Но большую 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)*. В этом случае весь трафик будет шифроваться, но надо принять еще некоторые меры для повышения безопасности.

Если у вас установлен Telnet-сервер, то попробуйте сейчас подключиться к нему, используя команду `telnet localhost`. Перед вами появятся следующие сообщения:

```
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:
```

Ничего страшного не замечаете? А я вижу подробную информацию о дистрибутиве и ядре. И все это становится известным еще до регистрации любому посетителю. Если хакер увидит открытый 23 порт, то ему уже не надо будет мучаться для выяснения, какая у вас ОС и версия ядра, достаточно подключиться к Telnet, и проблема решена.

Излишняя болтливость Telnet — это самая 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 указать ложную версию ядра, чтобы запутать хакера:

```
echo Debian Linux > /etc/issue
echo Kernel 2.4.4 on an i686 > /etc/issue.net
```

У меня установлен клон Red Hat дистрибутива с ядром 2.4.18-15asp, а хакер будет думать, что я использую Debian и старое ядро 2.4.4.

Проблема в том, что после перезагрузки содержимое файлов восстановится и Telnet снова в приветствии покажет всю информацию о системе. Чтобы этого не произошло, после изменения текста приветствия можно установить на файлы атрибут `-i`, который запрещает любые изменения:

```
chattr +i /etc/issue
chattr +i /etc/issue.net
```

5.1.6. 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 нужен для тестирования сервисов, то эти команды я включил в обзор только для того, чтобы вы удалили их из системы, и никогда не появлялся соблазн их использовать, и чтобы ими не смог воспользоваться хакер.

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. Этот вопрос стал актуальным только тогда, когда

начали происходить реальные взломы. 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.

Как мы уже говорили в *гл. 1*, при подключении по коаксиальному кабелю могут использоваться две схемы: все компьютеры объединяются напрямую в одну общую шину или кольцо, когда крайние компьютеры тоже соединены между собой (разновидность первого варианта). При использовании общей шины (рис. 5.2) все компьютеры подключены последовательно, и посылаемый пакет проходит по всем компьютерам, которые встречаются на пути, а установленные на них сетевые карты проверяют адресата: если пакет направлен им, то принять, иначе — пропустить.

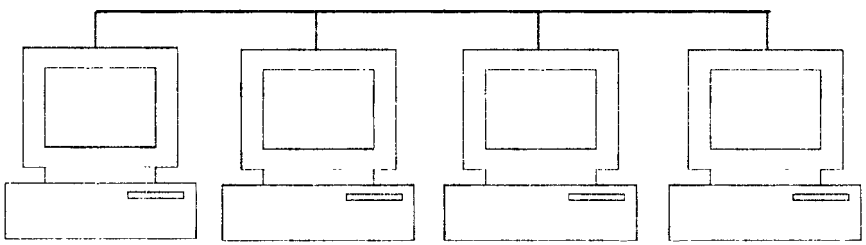

Рис. 5.2. Схема соединения компьютеров "Общая шина"

Компьютер обрабатывает только свои пакеты, но сетевая карта может видеть абсолютно все, что через нее проходит. И если очень сильно захотеть, то, воспользовавшись утилитой для прослушивания трафика (сниффер), можно и просмотреть все данные, проходящие мимо сетевой карточки, даже если они предназначены не вам. А т. к. большинство протоколов пропускают пакеты в открытом виде, то любой хакер может прослушать сеть и выявить конфиденциальную информацию, в том числе и пароли доступа.

Соединение по коаксиальному кабелю встречается все реже, потому что оно ненадежно и позволяет передавать данные максимум на скорости 10 Мбит/с. Да и сама схема подключения в общую шину не внушает доверия. При выходе из строя одного из компьютеров может нарушиться работа всей сети. Замыкание в кольцо отчасти снимает вопросы надежности, но не решает проблемы скорости и неудобства построения, обслуживания и использования такой сети.

При объединении компьютеров через центральное устройство хаб (Hub) или коммутатор (Swich) используется топология "Звезда" (см. рис. 5.3). В этом случае компьютеры с помощью витой пары получают одну общую точку. Такая схема надежнее и позволяет работать на скорости в 100 Мбит/с.

Если в центре стоит Hub (дешевое устройство), то пакеты, пришедшие с одного компьютера, копируются на все узлы, подключенные к этому хабу.

Таким образом, любой компьютер может видеть пакеты других участников сети.

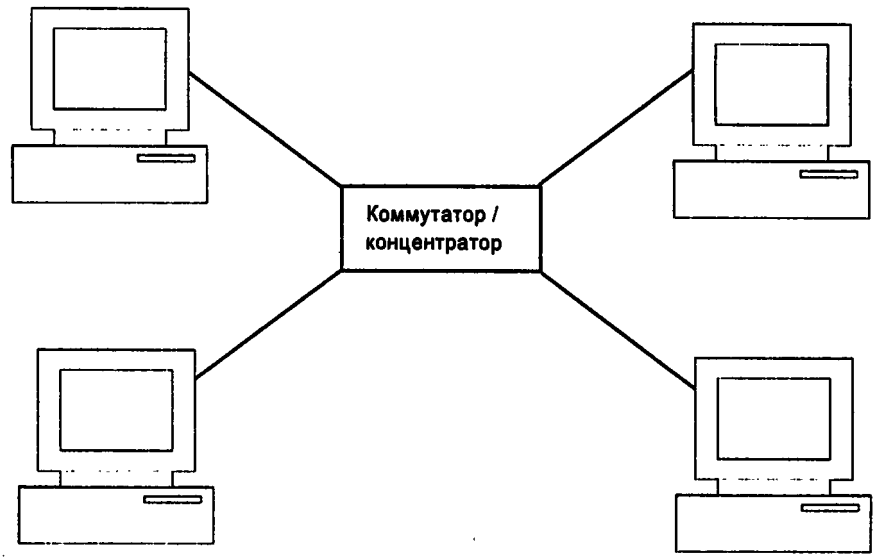

Рис. 5.3. Схема соединения компьютеров "Звезда"

В случае с коммутатором (более интеллектуальное устройство) пакеты будет видеть только получатель, потому что Swich имеет встроенные возможности маршрутизации, которые реализуются в основном на уровне MAC-адреса (Media Access Control Address, управление доступом к среде), который называют физическим адресом. Это 48-разрядный серийный номер сетевого адаптера, присваиваемый производителем. Он уникален, потому что у каждого изготовителя свой диапазон адресов. 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). В этом случае пакеты будут отправляться исходя из логических, а не физических адресов, и коммутатор сможет 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, потому что это накладно и в некоторых случаях просто невыполнимо. Но было найдено более удобное и универсальное решение — драйверы (tunneling), позволяющие программам

удаленного доступа различных разработчиков взаимодействовать друг с другом и поддерживающие несколько методов проверки подлинности, а также сжатия и шифрования данных. В общих чертах туннелирование (на примере FTP) выглядит следующим образом:

- на клиентском компьютере на определенном порту (условно Порт 1) вы должны запустить программу для шифрования трафика. Теперь, вместо того чтобы передавать данные на удаленный компьютер, вы должны с помощью FTP-клиента подсоединиться к Порту 1 своего компьютера и направлять данные на него, а уже там открытая программа их зашифрует и передаст по сети в закрытом виде;
- на удаленном компьютере на определенном порту запускается такая же программа, которая принимает зашифрованные данные, декодирует и передает их в открытом виде на порт FTP-сервера.

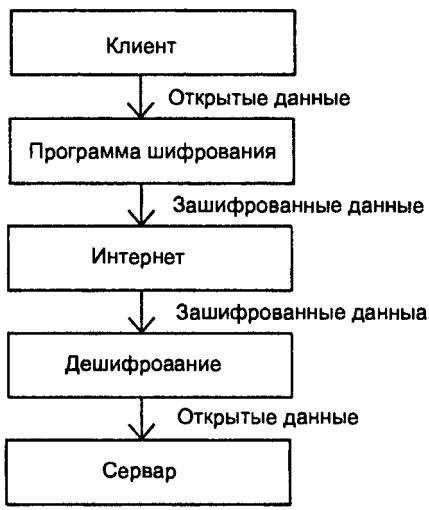

Рис. 5.4. Шифрование канала

На рис. 5.4 показано, как происходит шифрование данных. Получается, что все пакеты передаются через посредника, который кодирует данные. 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. Например, когда вы покупаете в электронном магазине какой-либо товар, то используется безопасное соединение с шифрованием, чтобы ни один злоумышленник не смог подсмотреть информацию о кредитной карте. В момент подключения к серверу браузер автоматически запускает на компьютере клиента программу шифрования и через

нее сервер передает зашифрованные данные и получает ответы в закодированном виде.

Получается, что шифрование не изменяет протокол, он остается тем же самым TCP/IP, просто на стороне клиента и сервера появляется посредник, который кодирует и декодирует данные. Использование такого метода очень удобно тем, что можно шифровать трафик любого протокола. Если в криптографическом алгоритме будет найдена ошибка или внесены качественные поправки, например, использование более длинного ключа, то не надо вносить изменения в сам протокол. Достаточно обновить программу и все новые возможности станут доступными.

Рассмотрим пример с Web-сервисом, который работает на 80 порту. На сервере можно запустить программу шифрования на 1080 порту, и все данные, которые будут приходить на него, будут декодироваться и передаваться на 80 порт. Если вы хотите предоставлять доступ к Web только по протоколу SSL, то 80 порт можно закрыть от вторжения извне сетевым экраном (о работе сетевого экрана мы подробно говорили в *гл.4*), а разрешить только подключения с сервера шифрования.

Адреса сайтов, которые защищены специальными ключами, начинаются с "https://". Буква "s" как раз и говорит о том, что соединение безопасно. Помимо этого, при подключении через обозреватель с включенным SSL в правом нижнем углу окна, в строке состояния появляется значок. Например, в популярном браузере Internet Explorer это пиктограмма висячего замка (рис. 5.5).

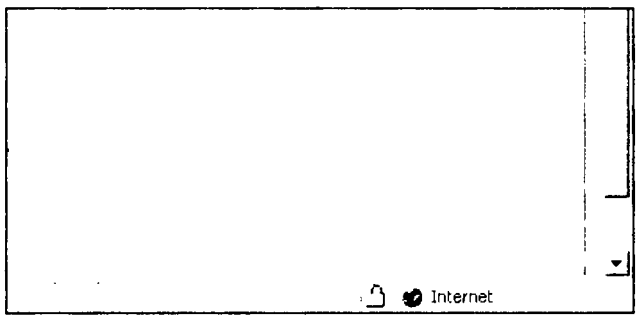

Рис. 5.5. Строка состояния Internet Explorer для безопасного соединения

Но этот опознавательный значок даже в Internet Explorer появляется не всегда. Более точно определить тип используемого соединения можно по свойствам документа. Большинство браузеров имеют команду просмотра свойств загруженной страницы, по вызову которой можно увидеть и используемое шифрование. Например, в Internet Explorer необходимо выбрать меню **File/Properties** (Файл/Свойства). Перед вами откроется окно, как на рис. 5.6.

Обратите внимание на поле **Connection**. Информация в нем свидетельствует, что используется протокол SSL 3.0 RC4 со 128-битной схемой шифрования.

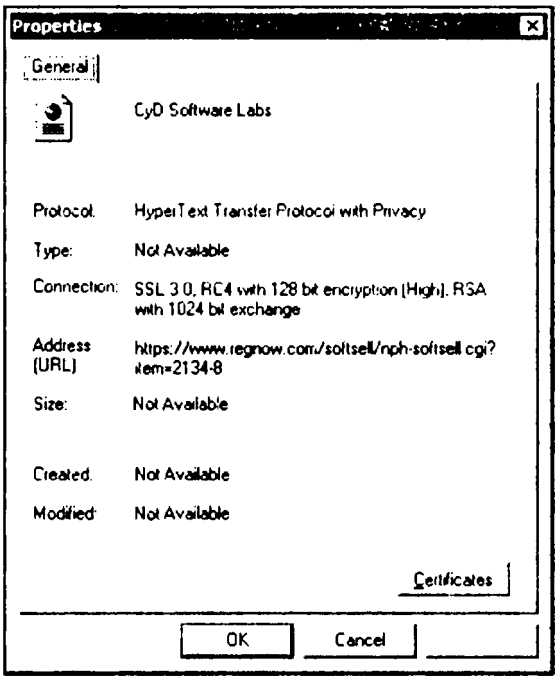

Рис. 5.6. Свойства документа в Internet Explorer

5.2.1. stunnel

В ОС Linux для шифрования и дешифрования трафика чаще всего используется программа stunnel. Но она только организует канал и выступает посредником, а для самого кодирования используется пакет OpenSSL, доступный в большинстве дистрибутивов Linux, и можно надеяться, что он у вас уже поставлен. Если нет, то это легко сделать с помощью установки соответствующего RPM-пакета с инсталляционного диска. Более подробную информацию по OpenSSL и 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 закодировать данные, но для расшифровки необходим закрытый ключ, который находится на клиенте.

У программ OpenSSL и stunnel очень много параметров, и рассматривать их все мы не будем. Лучше разберем реальный пример и познакомимся с наиболее часто используемыми аргументами.

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
```

В данном случае используется три параметра:

- `-p` — ключ, после которого указывается SSL-сертификат авторизации, по умолчанию уже созданный во время установки ОС или программы stunnel (находится 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, то будет прослушивание на всех интерфейсах локального компьютера. В качестве порта был выбран номер 9002. Все пришедшие на него данные считаются зашифрованными и будут декодироваться для передачи на другой порт локального компьютера;
- `-r` — после этого ключа указывается имя компьютера (не обязательный параметр) и порт, куда нужно передавать расшифрованные данные. Если хост не указан, то данные будут пересылаться на порт локального компьютера, в данном случае на 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.

Теперь рассмотрим запуск клиента. Тут все намного проще:

```
stunnel -c -d 1000 -r 192.168.77.1:9002
```

Здесь у нас появился один новый ключ:

- `-c` — означает, что туннель создается для клиента. По умолчанию программа stunnel запускается в серверном режиме.

Помимо этого, в нашем примере присутствуют 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 данном 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 закодированные данные будет принимать stunnel-сервер, расшифровывать и передавать их на 25 порт сервера.

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 контроля применяется 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` — на данном уровне сертификат является обязательным. Если сертификата нет или он не является подлинным, соединение не может быть установлено;
- `3` — наличие сертификата обязательно, а помимо этого, он должен быть в локальном хранилище (специальный список). В этом случае нужно указать директорию, в которой находятся сертификаты с помощью опции `-a`.

SSL-сервер может расшифровывать трафик и передавать на порт принимающей программы не только локального, но и другого компьютера. Таким образом, сервер SSL и сервер получатель трафика могут быть на разных компьютерах. Неплохо, чтобы сервер после расшифровки данных прятал IP-адрес клиента, с которого были отправлены данные, и это возможно, если указать опцию `-r`.

Во время установки пакета OpenSSL на вашем диске создаются сертификаты и пары ключей, которые используются для шифрования. Все это находится в директории **/usr/share/ssl/**.

С помощью опции `-n` можно непосредственно указать протокол, с которым будет происходить работа. В настоящий момент поддерживаются POP3 (Post Office Protocol, протокол обработки входящих сообщений), SMTP (Simple Mail Transfer Protocol, простой протокол электронной почты)) или NNTP (Network News Transfer Protocol, сетевой протокол передачи новостей).

Для большинства основных протоколов существуют номера портов, уже ставшие стандартом. Есть даже названия защищенных вариантов протоколов, которые, как правило, получаются за счет добавления к наименованию основного протокола буквы s, которая и указывает на безопасное соединение через SSL. Эта информация приведена в табл. 5.1.

Обратите внимание, что для протокола FTP требуется два защищенных канала. Один используется для управляющего соединения, а второй — для передачи данных. К этому мы еще вернемся в *гл. 10*, когда будем рассматривать этот протокол.

Таблица 5.1. Список протоколов с номерами портов

Протокол	Название SSL варианта протокола	Номер TCP-порта
HTTP	HTTPS	443
SMTP	SMTPS	465
LDAP	LDAPS	636
TELNET	TELNETS	992
SHELL	SSHELL	614
FTP	FTPS	990
FTP-DATA	FTPS-DATA	989
IMAP	IMAPS	993
POP3	POP3S	995
IRC	IRCS	994

5.2.3. Шифрование файлов

Некоторые серверы могут использоваться для хранения архивных данных, которые, несмотря на такой статус, должны быть скрыты от стороннего взгляда. Наилучший вариант защиты — шифровать файлы, чтобы никто не смог увидеть их содержимое, и пакет OpenSSL предоставляет нам такую возможность.

Шифрование необходимо не только для резервных копий файлов или архивных данных, но и для файлов с секретной информацией, которые необходимо передать по незащищенным каналам связи, например, через E-mail-почту или публичный FTP-сервер.

Для шифрования необходимо выполнить команду `/usr/bin/openssl`, которая имеет следующий вид:

```
/usr/bin/openssl алгоритм -in файл1 -out файл2
```

Количество используемых алгоритмов исчисляется десятками. Наиболее распространенным является DES (Data Encryption Standard, стандарт шифрования данных). Я тоже отдаю предпочтению 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 необходимо добавить ключ -d. Помимо этого, в качестве -in задается закодированный файл, а в параметре -out — имя файла для сохранения результата:

```
/usr/bin/openssl алгоритм -d -in файл2 -out файл1
```

Рассмотрим пример шифрования всего списка паролей **/etc/passwd** в файл **/home/passwd** по алгоритму DES. Для этого выполняем команду:

```
/usr/bin/openssl des -in /etc/passwd -out /home/passwd
```

В ответ на эту директиву программа попросит вас указать пароль и затем подтвердить ввод, дабы исключить возможные ошибки.

Выполните команду `cat /home/passwd` и убедитесь в том, что содержимое этого файла не читаемо.

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 *гл. 13*, которая будет полностью посвящена этому вопросу.

5.2.4. Туннель глазами хакера

Хакеры могут использовать туннели для своих целей. Например, совсем недавно я подключился к Интернету по технологии ADSL (Asymmetric Digital Subscriber, асимметричная цифровая абонентская линия). В абонентскую плату входит всего 400 Мбайт трафика. Ну что такое 400 Мбайт? Для меня это неделя работы в экономном режиме, потому что общаться приходится много, а из почтового ящика я иногда выкачиваю до 20 Мбайт в день. Оплата сверх абонентской платы обходится достаточно дорого.

Как можно обойти ограничение и получить бесплатный трафик? Подобно большинству провайдеров, мой предоставляет абсолютно бесплатный трафик со своих серверов, к которым можно обращаться сколько угодно. Жаль, что на этих серверах ничего полезного нет, но это легко исправить.

В мою абонентскую плату входит 10 Мбайт серверного пространства для создания визитной карточки в Интернете. Максимум, что можно там разместить, — домашнюю страничку. Но мне это не нужно. Главное, что трафик для этого сайта неограничен.

Итак, если установить на сервер программу туннелирования, то можно организовать соединение, как показано на рис. 5.7.

Хакер подключается через программу stunnel к априори бесплатному Web-серверу, который находится на площадке провайдера. Оттуда весь трафик

перенаправляется на какой-нибудь прóкси-сервер в Интернете или напрямую передается Web-сайтам. За это также вносить деньги не надо, потому что связь с Web-сервером в обе стороны бесплатна.

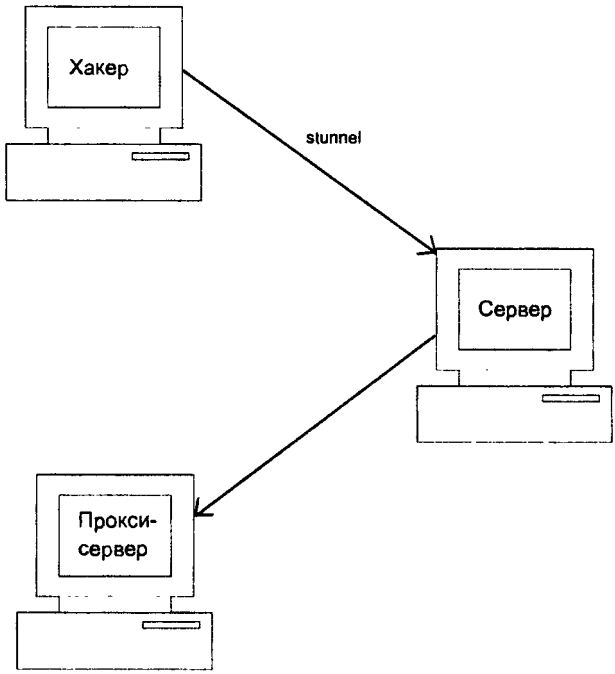

Рис. 5.7. Подключение к Интернету через Web-сервер

Таким образом можно скачивать не 400 Мбайт трафика, а целые гигабайты, и все на халяву. Если вы решили использовать этот метод, то не забудьте, что на Web-сервер необходимо повесить хоть какую-то страничку. Иначе администратор моментально заметит, что идет большой трафик на пустую Web-страницу и сможет вычислить туннель.

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 только обращение к Web-сайтам по протоколу HTTP. Все остальное было воспрещено. Руководство мотивировало это тем, что пользователи не должны иметь возможность передачи файлов в Интернет.

Как можно обойти такое ограничение? Снова ставим туннель на Web-сервере, и можно использовать любой другой протокол, пряча весь трафик в

HTTP, откуда туннель уже перенаправляет данные на нужные порты с необходимым протоколом.

Например, нам нужен доступ к FTP. На каком-нибудь сервере в Интернете на 80 порту, доступ к которому разрешен, потому что здесь должен находиться Web-сервис, ставим сервер туннеля и настраиваем его на подключение к нужному FTP-серверу. А на своем компьютере устанавливаем клиента для соединения с 80 портом 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. Можно даже запрятать нужный протокол в DNS- или ICMP-пакеты, если они разрешены в вашей сети. Если ICMP заблокировать можно, то с DNS это практически нереально, т. к. при подключении к Интернету без DNS-службы работать сложно.

Как видите, абсолютно безопасная на первый взгляд и даже предназначенная для защиты технология превращает в средство взлома ограничения, которыми наградил вас администратор.

Если вы обладаете хорошими знаниями программирования на PHP или Perl, то можете написать Web-сценарии, которые на сервере будут обращаться к необходимым вам ресурсам. Получится общение с необходимым сервером через Web-браузер, это как работа с почтой через Web. Эта возможность есть на большинстве почтовых сервисов, и вам она должна быть знакома на практике.

5.3. Протокол SSH

Мы уже говорили о том, что протокол Telnet не очень хорошо подходит для удаленного управления сервером, потому что далеко не безопасен. А желание и потребность в этом есть. В больших сетях, как правило, используются несколько серверов, и бегать от одного монитора к другому накладно и неудобно. Любой администратор хочет сидеть за одним компьютером и управлять всем комплексом одновременно, и при этом по безопасным каналам связи.

Во время таких сеансов администратор передает по сети много конфиденциальной информации (например, пароли root), которая ни в коем случае не должна быть видна прослушивающим утилитам. Для обеспечения безопасной связи создано множество различных программ, но самой популярной стала SSH, которая сейчас поставляется в большинстве дистрибутивов Linux.

Теперь вы можете спокойно управлять своей сетью, удаленно подключаться к серверам, и нет необходимости на каждом из них держать мониторы. У ме-

ня сделано именно так (экономия на железе), и есть только один дежурный монитор, который я могу подсоединить к любому системнику, если надо решать проблему не по сети.

Преимущество SSH заключается в том, что этот протокол позволяет удаленно выполнять команды, но при этом требует аутентификацию и шифрует канал связи. Важным моментом является то, что даже пароли при проверке подлинности пользователя передаются в зашифрованном виде.

На данный момент 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 на данный момент поддерживаются обе версии.

Итак, в ОС Linux за протокол SSH отвечает программа OpenSSH, для которой родной платформой можно считать другую Unix-систему — OpenBSD, а впоследствии сервис клонировался на 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 серверной и клиентской частей. 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 директории **/etc/ssh**. Здесь можно увидеть следующий перечень:

- 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 крипто-

стойких алгоритма DSA (ssh_host_dsa_key, ssh_host_dsa_key.pub) и RSA (это сокращение состоит из имен основателей алгоритма шифрования: Ronald Rivest, Adi Shamir и Leonard Adleman) (ssh_host_rsa_key и ssh_host_rsa_key.pub). Оставшиеся два файла ssh_host_key, ssh_host_key.pub хранят ключи для первой версии SSH. Для каждого алгоритма требуется по два файла: с расширением pub — хранит открытый ключ, без расширения — содержит приватный ключ.

С помощью открытого ключа можно закодировать данные и отправить на сервер, но для расшифровки нужен только закрытый ключ, который не может быть подобран простыми алгоритмами, он должен быть 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). Его можно увидеть в листинге 5.1. Файл небольшой, поэтому для удобства я привел его полностью, убрав только некоторые комментарии.

Листинг 5.1. Файл конфигурации sshd

```
#Port 22
#Protocol 2,1
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey for protocol version 1
#HostKey /etc/ssh/ssh_host_key
#HostKeys for protocol version 2
#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_dsa_key

#Lifetime and size of ephemeral version 1 server key
#KeyRegenerationInterval 3600
#ServerKeyBits 768

#Logging
#obsoletes QuietMode and FascistLogging
#SyslogFacility AUTH
SyslogFacility AUTHPRIV
#LogLevel INFO
```

```
#Authentication:

#LoginGraceTime 600
#PermitRootLogin yes
#StrictModes yes

#RSAAuthentication yes
#PubkeyAuthentication yes
#AuthorizedKeysFile        .ssh/authorized_keys

#rhosts authentication should not be used
#RhostsAuthentication no
#Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes
#For this to work you will also need host keys in
/etc/ssh/ssh_known_hosts
#RhostsRSAAuthentication no
#similar for protocol version 2
#HostbasedAuthentication no
#Change to yes if you don't trust ~/.ssh/known_hosts for
#RhostsRSAAuthentication and HostbasedAuthentication
#IgnoreUserKnownHosts no

#To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no

#Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes

#Kerberos options
#KerberosAuthentication automatically enabled if keyfile exis
#KerberosAuthentication yes
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes

#AFSTokenPassing automatically enabled if k_hasafs() is true
#AFSTokenPassing yes

#Kerberos TGT Passing only works with the AFS kaserver
#KerberosTgtPassing no

#PAMAuthenticationViaKbdInt yes

#X11Forwarding no
X11Forwarding yes
#X11DisplayOffset 10
```

```
#X11UseLocalhost yes
#PrintMotd yes
#PrintLastLog yes
#KeepAlive yes
#UseLogin no

#MaxStartups 10
#no default banner path
#Banner /some/path
#VerifyReverseMapping no

#override default of no subsystems
Subsystem       sftp    /usr/libexec/openssh/sftp-server
```

Рассмотрим основные параметры, которые вам могут пригодиться:

- `Port` — порт, на котором ожидаются подключения. По умолчанию это 22. Некоторые администраторы любят изменять это значение, перенося сервер на другой порт. В какой-то степени это оправдано. Например, если у вас нет Web-сервера, то можно поместить SSH на 80 порт. Хакеры будут думать, что это Web-сервер, и не будут его ломать;
- `Protocol` — поддерживаемые протоколы. Обратите внимание, что первым идет число 2, а затем 1. Это значит, что сервер будет сначала пытаться подключиться по протоколу второй версии, и только потом по первому. Я рекомендую убрать комментарии 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` — определяет адреса для прослушивания подключения. У вашего сервера может быть несколько сетевых карт. По умолчанию идет прослушивание всех интерфейсов. Вы должны указать только те, с которых вы 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 только из внутренней сети, то следует прослушивать только этот адрес (задается в формате `адрес:порт`). Разрешается описывать несколько таких записей, чтобы указать требуемые интерфейсы;
- `HostKey` — указывается путь к файлам, содержащим ключи шифрования. Нужно прописывать только закрытые ключи, которые использует сервер для расшифровки пакетов;
- `KeyRegenerationInterval` — в версии 1 во время сессии могут регенерироваться ключи. Это позволяет 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 1 версии протокола (см. параметр `protocol`), то этот атрибут не влияет на работу;

- `ServerKeyBits` — длина серверного ключа. По умолчанию установлено `768`, минимальное значение — `512`;
- `SyslogFacility` — задает 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` — определяет, разрешено (`yes`) или запрещено (`no`) входить в систему по SSH-протоколу 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` — указывает, должен ли sshd проверять состояние файлов и их владельцев, пользовательских файлов и домашней директории до ввода пароля. Желательно установить `yes`, потому что многие начинающие пользователи делают свои файлы всеобщими для записи;
- `RSAAuthentication` — разрешена ли аутентификация по алгоритму RSA. Параметр действует для 1 версии протокола;
- `PubkeyAuthentication` — дозволена ли аутентификация по публичному ключу. Параметр действует для 2-й версии протокола;
- `AuthorizedKeysFile` — файл с публичным ключом, который может использоваться для аутентификации;
- `RhostsAuthentication` — разрешение аутентификации 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` — если параметр равен `yes`, то запрещается 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 — если этот параметр yes, то при аутентификации будет требоваться ключ хоста из директории **/etc/ssh/ssh_known_hosts**. Параметр используется в 1 версии SSH;
- IgnoreUserKnownHosts — если параметр равен no, то необходимо доверять компьютерам из списка **~/.ssh/known_hosts** при RhostsRSAAuthentication-аутентификации. Не верьте никому, поэтому параметр лучше всего изменить на yes;
- PasswordAuthentication — если значение равно yes, то будет требоваться пароль. При использовании авторизации через ключи параметр можно отключить;
- PermitEmptyPasswords — по умолчанию установлено no, что запрещает использование пустых паролей, и изменять это значение не стоит;
- KerberosAuthentication — использовать проверку 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 — очищать билет Kerberos из кэша при выходе из системы;
- Banner — позволяет указать файл, в котором находится текст приветствия, отображаемый пользователями.

5.3.3. Параметры доступа к серверу sshd

Кроме приведенных в листинге 5.1 можно использовать следующие директивы:

- AllowGroups — позволить вход в систему только пользователям указанных групп (перечисляются через пробел в одной строке);
- AllowUsers — разрешить вход в систему пользователям, имена которых перечислены через пробел;
- DenyGroups — запретить вход в систему пользователям указанных через пробел групп;
- DenyUsers — запретить вход в систему пользователям, имена которых перечислены 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 директории пользователя. В листинге 5.2 приведено содержимое глобального файла настроек без некоторых комментариев.

Листинг 5.2. Конфигурационный файл /etc/ssh/ssh_config

```
# Site-wide defaults for various options

# 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,aes192-
cbc,aes256-cbc
#   EscapeChar ~
Host *
Protocol 1,2
```

Некоторые из этих параметров мы уже видели при рассмотрении серверного конфигурационного файла. Здесь также присутствует параметр `Protocol`, в котором указываются используемые версии SSH. Только в данном случае не стоит запрещать 1 версию. 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 ☺.

Перечислю характерные для клиента команды:

- `Host` — указывает, к какому серверу будут относиться следующие настройки;
- `CheckHostIP` — разрешена проверка IP-адреса с перечисленными в файле know_hosts адресами;
- `Compression` — позволяет (`yes`) или запрещает (`no`) использование сжатия данных;
- `KerberosAuthentication` — позволяет (`yes`) или запрещает (`no`) использование аутентификации по протоколу Kerberos;
- `NumberOfPasswordPrompts` — задает количество попыток ввода пароля. Если пароль не введен верно, то соединение разрывается;
- `IdentityFile` — определяет имя файла, содержащего закрытые ключи пользователя;
- `PasswordAutentication` — указывает на использование аутентификации по паролю.

5.3.5. Пример работы клиента SSH

Теперь рассмотрим на примере, как можно подключиться к удаленному серверу. Для этого используется команда:

```
ssh пользователь@сервер
```

Например, чтобы подсоединиться к серверу fltnovm 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)?
```

Данным 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 вам сообщается, что ключ добавлен в список известных хостов. Это значит, что в вашей домашней директории, в папке **.ssh/** 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 вы сидите за его клавиатурой.

5.3.6. Вход по ключу

Намного удобнее и даже безопаснее способ авторизации по ключу, а вход по паролю может быть даже заблокирован. Обращение к системе по SSH не совсем 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?

Для каждого подключения должны быть свои пароли. Но помнить их все очень сложно, поэтому лучше для авторизации использовать ключи, которые итак защищены, дальше некуда. Нужно сделать только небольшие изменения в конфигурации.

Для начала нужно создать новый ключ. Для этого используется программа ssh-keygen. Ей нужно передать два параметра:

- `-t` — тип ключа. Здесь можно указывать `rsa` или `dsa` для второй версии SSH или `rsa1` — для первой. Для примера будем использовать `rsa` ключ;
- `-f` — файл, в котором будет сохранен закрытый ключ. Открытый ключ получит такое же имя, но с расширением pub;
- `-b` — длина ключа, которая может быть минимально `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 директории .ssh — своей домашней директории (об этом свидетельствует знак "~"). Это директория, в которой SSH будет искать все настройки. Если вы еще не подключались к серверу, то этот путь и ключ отсутствуют. Для исправления ситуации нужно перейти в свою домашнюю директорию и создать папку .ssh:

```
cd /home/flenov
mkdir .ssh
```

Если при генерации ключа не указывать файл для его сохранения, то по умолчанию он будет создан в директории **~/.ssh/** с именем id_rsa для RSA-шифрования. Для DSA-шифрования файл будет располагаться там же.

но с именем 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):
```

В данном сообщении говорится, что начат процесс генерации публичного и закрытого RSA-ключей. Вам предлагается ввести пароль или оставить его пустым. Я рекомендую лучше указать достаточно длинный пароль (не менее 10 символов, а лучше фразу). 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 первой строке нас проинформировали о том, что закрытый ключ сохранен в файле **~/ssh/myrsakey**, а открытый — в **~/ssh/myrsakey.pub**.

Получив ключи, вы должны отправить файл **~/ssh/myrsakey.pub** на удаленный компьютер, чтобы SSH-сервер мог использовать его для аутентификации. Для передачи можно смело использовать открытые каналы связи, потому что публичный ключ ничего не стоит без фразы, которую вы ввели, и без секретного ключа. Даже если хакер сможет получить файл myrsakey.pub, пользы от этого 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, 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 будет устанавливаться только на основании ключей. Для обеспечения безопасной связи этого достаточно.

5.3.7. X11 в терминале

Использование командной строки для управления удаленной системой позволяет значительно сэкономить трафик. Но иногда нужен графический режим. Я не рекомендую его использовать в целях безопасности и для повышения производительности, но пользователи Windows могут просто не смириться с командной строкой. Если вы любите красивые окна, то SSH-сервер может перенаправить X11 (графическую оболочку ОС Linux) на ваш терминал. Для этого в конфигурационном файле sshd_config должны быть указаны следующие три директивы:

- `X11Forwarding yes` — разрешает переправлять ОС Linux графический режим X11;
- `X11DisplayOffset 10` — первый номер дисплея, доступный для SSH-сервера. По умолчанию `10`, и это значение можно так и оставить;
- `X11UseLocalhost yes` — если параметр равен `yes`, то в качестве X-сервера будет использоваться локальный. В этом случае клиент будет работать с нашим X11, а служебная информация, передаваемая по сети, будет шифроваться.

Если вы хотите подключаться к графической оболочке Linux из Windows, то вам понадобится программа типа X11 для этой ОС. В качестве примера могу порекомендовать X–Win32-клиент, который можно скачать с сайта **http://www.starnet.com/**.

Я не рекомендую использовать X11, потому что эта технология отлажена еще очень плохо, и существуют методы подделки и взлома соединения.

5.3.8. Защищенная передача данных

В состав пакета SSH входят еще две полезные программы — это sftp-server (FTP-сервер с поддержкой шифрования передаваемых данных) и sftp (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 данной строке запускается sftp-server из пакета OpenSSH.

Работа с клиентом sftp не отличается от работы SHH-клиента. Выполните команду `sftp localhost` и перед вами появится приглашение авторизации, которую мы рассматривали в *разд. 5.3.5*. Введя правильный пароль, вы оказываетесь в командной строке FTP-клиента и можете передавать или принимать файлы, используя команды FTP-протокола. Этот протокол мы будем подробно рассматривать в *гл. 10*, а сейчас вам достаточно знать, что большинство команд схожи с директивами Linux для управления файлами.

Попытайтесь сейчас воспользоваться клиентом sftp для подключения к своей системе. Авторизовавшись, можно попробовать выполнить команды `ls` или `cd`, чтобы убедиться в работоспособности программы. Для выхода из sftp наберите команду `exit`. Основные команды FTP-протокола можно увидеть в *приложении 1*.

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.

Вы должны учитывать, что далеко не все серверы и клиенты FTP поддерживают шифрование SSH, поэтому убедитесь в поддержке этого протокола со стороны вашего программного обеспечения.

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 зачем постоянно держать программу в памяти, особенно если она слишком большая, а работает очень редко. Намного лучше, когда один сервис в системе будет следить за портами, и если пользователь обращается к определенному каналу, то эта программа запустит необходимый сервис. В ОС Linux такая возможность есть, и для этого используется демон inetd или более новая версия — xinetd.

Как определить, что запускать? Для этого используется файл **/etc/services**, в котором находится список сервисов и их портов в следующем формате:

```
имя порт/протокол псевдоним
```

- `имя` — название сервиса, который необходимо запускать;
- `порт` — номер канала, который должен прослушиваться;

- протокол — сервис inetd умеет работать с TCP- и UDP-протоколам, порты которых не пересекаются (и они совершенно разные), то необходимо в явном виде указывать протокол;
- псевдоним — вымышленное имя для сервиса.

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 такие строки, чтобы вы увидели варианты описания различных сервисов.

Если вы используете старый дистрибутив ОС Linux, то в нем, скорей всего, еще работает inetd. Но мы уже говорили, что давнишний дистрибутив не может быть безопасным, и с ним что-либо делать бесполезно. Лучший защита — обновление, и тогда основным сервисом станет xinetd, который становится, если уже не стал, стандартом для всех.

Я рекомендую переход на xinetd, потому что в нем появилось много дополнительных возможностей, которые делают удобным администрирование и повышают безопасность. Например, в сервис xinetd встроены проверка всех удачных и неудачных соединений, возможность контроля доступа и даже предоставление его строго в определенное время.

5.4.1. Конфигурирование xinetd

Основной конфигурационный файл для xinetd — это **/etc/xinetd.conf**. В нем описываются настройки по умолчанию для запускаемых сервисов и директория, в которой будут находиться конфигурационные файлы, влияющие на работу конкретных сервисов. Рассмотрим по листингу 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 директории /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 директорию **/etc/xinet.d**:

```
includedir /etc/xinetd.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 telnet-сессии.
# Он использует не зашифрованные 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` — если этот параметр установить в `true`, то сервис будет запрещен для исполнения;
- `flags` — атрибуты выполнения сервиса;
- `socket_type` — тип используемого сокета. Для протокола TCP здесь должно быть значение `streem`, 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` — запись событий будет производиться в указанный файл или системный журнал;
- `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 юзеры 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? Не имея прав root, пользователь не сможет запустить сервер, который работает с портом от 1 до 1024. Такая защита необходима, потому что в данном диапазоне функционируют очень важные сервисы, и их нельзя запускать простому пользователю.

Представьте себе, что хакер с правами простого пользователя смог бы запустить FTP-сервер, который используется для передачи файлов. Сделав это, он

получит 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 команды: `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, не должны иметь доступ все компьютеры с адресами 192.168.1.X, где X может быть любым числом. В этом случае нужно 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.

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 прописывать 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 использовать 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 из этой сети с номером 192.168.1.1 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 данном случае я бы увеличил второе значение до 19:00, потому что очень часто сотрудники 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 повторяют 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.

5.4.3. Недостатки xinetd

У любой технологии есть недостатки, и initd/xinitd не являются исключением. При подключении пользователя к одному из портов вашего сервера программа initd (или xinitd) просматривает таблицу портов, чтобы найти сервис, который необходимо запустить, и передать ему управление. Это может повлечь за собой большие затраты на поиск сервиса и его запуск.

В идеальном случае, любой запрос должен выполняться максимально быстро, иначе это грозит нам увеличением времени отклика. Но не это самое страшное. Лишнюю пару секунд пользователь может подождать, а вот хакер ждать не будет. Он может направить большое количество запросов на соединение с сервером, и программа initd (или xinitd) съест все ресурсы своим поиском и запуском сервисов. Таким образом, увеличивается вероятность удачного проведения DoS атаки со стороны хакера.

ГЛАВА 6

В стиле Samba

Изначально для обмена файлами между компьютерами использовался FTP-протокол (об этом подробно поговорим в *гл. 10*). Но он неудобен, т. к. использует технологию "клиент-сервер". Чтобы получить с компьютера друга файл, он должен запустить у себя FTP-сервер, а вы с помощью специальной программы FTP-клиента должны подключаться к этому серверу и скачивать нужный файл.

Чтобы не утруждать себя настройками FTP-сервера, многие стали использовать специализированные обменники в своих локальных сетях. Для этого выделялся FTP-сервер с открытым доступом, который превращался в мусорную корзину.

В Windows появился более удобный способ публиковать свои файлы — "Сетевое окружение", с помощью которого 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 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 (чаще всего 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.

С помощью Samba можно сделать легко доступный и удобный в использовании файловый сервер. Раньше у меня на работе для этих целей использовался Windows 2000-сервер, который параллельно работал как сервер баз данных. Но файловый архив занимает слишком много места, отнимает ресурсы сети и сервера и понижает безопасность. Поэтому было принято решение перенести файловый архив на отдельный физический сервер. Использовать для этого

Windows 2000 слишком дорого и неэффективно — это то же самое, что вывозить мусор из дома на расстояние 20 метров на Ford Mondeo. Для этих целей лучше подойдет Linux, который обходится дешевле и не требует обслуживания. Достаточно один раз настроить такой сервер, и можно использовать годами. Именно так мы и поступили.

Одно из мощнейших преимуществ Samba — возможность удаленного управления через SSH или SWAT (Samba Web-based Administrative Tool, набор администратора через Web для Samba).

6.1. Конфигурирование Samba

Основным конфигурационным файлом для Samba является smb.conf, который можно найти в директории **/etc/samba** (в некоторых дистрибутивах это может быть каталог **/etc**). Кроме него в этой директории можно найти файл lmhosts, с помощью которого происходит преобразование IP-адресов и имен компьютеров аналогично **\etc\hosts** в Linux, а **Диск:\WINDOWS\system32\drivers \etc\lmhosts.sam** в Windows только используется сервером Samba.

Дополнительно вы можете создать следующие файлы (некоторые из них могут существовать):

- smbusers — файл хранит 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 исключительно root, и никто более.

Конфигурационный файл smb.conf содержит не так много директив, поэтому для удобства восприятия я привел небольшой пример (листинг 6.1), 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

# Директивы безопасности
   security = user
;   password server = <NT-Server-Name>
;  password level = 8
;  username level = 8
   encrypt passwords = yes
   smb passwd file = /etc/samba/smbpasswd

   unix password sync = Yes
   passwd program = /usr/bin/passwd %u
   passwd chat = *New*password* %n\n *Retype*new*password* %
*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=819
;   interfaces = 192.168.12.2/24 192.168.13.2/24

# 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 server = w.x.y.z
;   wins proxy = yes
    dns proxy = no

# 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 директорий. Я все это удалил, чтобы вам проще было ориентироваться, когда мы будем рассматривать назначение команд.

В большинстве конфигурационных файлов Linux и его программах для описания директив используется формат:

```
ИмяДирективы Значение
```

Имя директивы в данном случае должно состоять из одного слова и не может содержать пробелы. После имени ставится пробел, за которым идет значение директивы.

В Samba-сервере используется несколько иной формат:

```
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).

6.1.1. Основные настройки

Конфигурационный файл разбит на секции. Самой первой идет [global], в которой описываются глобальные настройки сервера. В ней можно увидеть следующие директивы:

- workgroup = имя — содержит имя группы, в которую входит сервер. Когда в Windows вы входите в сетевое окружение, то можно увидеть все доступные ресурсы, разбитые на категории. В каждой группе могут быть свои компьютеры или серверы;
- netbios name = имя — определяет имя, которое пользователи будут видеть в сетевом окружении для данного сервера, оно не должно совпадать с именем рабочей группы;

- `server string = описание` — это 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` — перечисление через пробел IP-адресов или сетей, которым разрешен доступ к Samba-серверу. Например, чтобы открыть доступ всем компьютерам сети 192.168.1.x и одному компьютеру из другой сети 192.168.2.1, надо написать следующую команду:

```
hosts allow = 192.168.1. 192.168.2.2
```

- `printcap name = файл` — указывает файл описания принтеров, подключенных к системе. По умолчанию это **/etc/printcap**;
- `load printers = yes` — задает (yes) автоматическое включение принтеров в список открытых ресурсов. Если в этом нет надобности, то укажите `no`;
- `printing = система` — определяет тип системы печати. Здесь можно использовать одно из следующих значений: `bsd`, `sysv`, `plp`, `lprng`, `aix`, `hpux` или `qnx`.

6.1.2. Безопасность

В этом разделе мы рассмотрим директивы, которые напрямую или косвенно влияют на безопасность:

- `guest account = имя` — указывается учетная 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 не безопасен;
- `log file = файл` — название файла-журнала. например, можно написать **/var/log/samba/%m.log**. Обратите внимание, что в имени присутствует символ процента, вместо которого во время работы будет подставляться имя 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;
- `security = уровень` — определяет степень доступа, параметр может принимать одно из следующих значений:
  - `user` — на уровне пользователя;
  - `share` — аутентификация на основе имени и пароля;

- `server` — задает имя сервера, на котором хранятся пароли (с помощью директивы `password server = ИмяСервера`), т. к. пароли держатся на другом smb-сервере;
- `security = domain` — включает сервер в домен Windows NT, пароль для доступа указывается в файле, определенном с помощью директивы `smb passwd file = ИмяФайла`;

❐ `encrypt passwords = yes` — позволяет шифровать пароли, передаваемые по сети. Данная опция требует пояснений, потому что могут возникнуть проблемы при авторизации с компьютеров с Windows-системой.

Суть в том, что Windows так шифрует пароли, что они могут быть восстановлены. Зашифрованный пароль передается по сети, и на сервере происходит его дешифровка. Linux же использует необратимую схему хэширования по алгоритму MD5. Для проверки правильности пароля он шифруется по тому же алгоритму, а затем сравнивается результат (хэш-сумма). Таким образом, шифрование Linux и Windows не совместимы.

Чтобы пользователи Windows смогли авторизоваться на Samba-сервере, необходимо передавать пароль в незашифрованном виде. Для этого значение `encrypt passwords` должно быть равно `no`. При этом в Windows-системе необходимо в реестре 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\Parametrs

В Windows 2000:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanWorkStation\Parameters

Если соответствующего параметра не существует, то 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 используете шифрованный пароль, то при авторизации будет использоваться файл **/etc/samba/smbpasswd** (можно изменить с помощью

директивы `smb passwd file`). Он необходим из-за отличий в шифровании Windows и Linux.

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 расположения файла с паролями. По умолчанию это та же директория, где расположены конфигурационные файлы Samba;
- `ssl CA certFile = файл` — указывает на 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 нет такой необходимости, то установите значение параметра `no`. Для работы этой директивы нужно указать программы, которые будут изменять пароли (команда `passwd program`) и сообщения, появляющиеся перед пользователем (команда `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 Samba-сервера для Windows 9*x*-клиентов (о нем подробнее мы поговорим в *разд. 6.3*).

6.1.3. Сеть

В этом разделе нам предстоит рассмотреть директивы настройки сетевого протокола:

- `include = файл` — позволяет подключить конфигурационный файл smb.conf с другого компьютера. Имя файла задается в формате `путь.%m`. В данном случае `путь` — это полное имя файла на удаленной машине, а `%m` — это 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 буфера.

В данном случае:

- TCP_NODELAY — позволяет быстрее передавать данные (без задержек);
- SO_RCVBUF — размер принимающего буфера;
- SO_SNDBUF — размер передающего буфера;

- interfaces = интерфейсы — если у вас установлено две сетевые карты, которые направлены на разные сети, то с помощью этой директивы можно позволить работать с Samba пользователям из обеих сетей.

6.1.4. Как домен

Если проанализировать все параметры, которые мы будем рассматривать в этом разделе, то станет очевидно, что Samba способна заменить Windows-сервер, и рабочие станции на базе Windows не заметят неудобств:

- local master = yes — позволяет сделать ваш Samba-сервер основным браузером сети;
- domain master = yes — позволяет сделать Samba-сервер главным браузером домена. Не устанавливайте значение yes, если в вашей сети есть Windows NT контроллер домена;
- domain logons = yes — если включить эту опцию, то Samba сможет работать как сервер для входа в систему компьютеров с ОС Windows 95. При загрузке компьютера с Windows можно будет вводить пароли из Samba;
- logon script = файл — если директива domain logons включена, то в этой команде можно указать файлы bat-сценариев, которые будут выполняться на компьютере клиента при входе в систему. Сценарий может задаваться в виде %m.bat (заменяется именем компьютера) или %U.bat (подменяется на имя пользователя);
- logon path = путь — определяет место хранения пользовательских профилей. Для использования этой директивы необходимо убрать комментарии с секции [Profiles], которую можно найти в файле конфигурации по умолчанию.

6.1.5. Поддержка WINS

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 использует доменные имена.

Для настройки работы через WINS используются следующие директивы:

- `wins support = yes` — разрешить использование WINS-сервера;
- `wins server = w.x.y.z` — адрес WINS-сервера;
- `DNS Proxy = yes` — если эта опция разрешена, то не распознанные NetBIOS-имена можно попытаться определить через DNS-сервер.

6.1.6. Отображение файлов

В Linux и Windows используются 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. Для решения этой проблемы есть несколько команд:

- `case sensitive = no` — позволяет не принимать во внимание чувствительность к регистру;
- `default case = lower` — отображать все файлы в нижнем регистре;
- `preserve case = no` и `short preserve case = no` — эти директивы позволяют сохранять информацию с учетом регистра.

Если у вас в сети работают пользователи Windows-систем, то рекомендуется оставить указанные выше значения. Для однородной Linux-сети можно разрешить сохранять регистр.

6.2. Описание объектов

После того как вы определили основные параметры Samba-сервера, можно описывать объекты, к которым может получать доступ пользователь. Эти делается в отдельных разделах, которые идут после секции `[Global]`, которую мы рассмотрели в *разд. 6.1*.

6.2.1. Пора домой

Конечно же, любой пользователь захочет работать со своей директорией. Для этого он должен иметь учетную запись в Linux, с которой и будет связан его каталог. Обращение к такой директории будет выглядеть как `//сервер/имя`, где `сервер` — это имя сервера или его IP-адрес, а `имя` — это имя пользователя, домашнюю директорию которого необходимо увидеть.

Для того чтобы разрешить пользователям работать с собственными директориями, нужно описать раздел `[homes]`. Рассмотрим пример такой секции:

```
[homes]
  comment = Home Directories
```

```
browseable = no
writable = yes
valid users = %S
create mode = 0664
directory mode = 0775
```

В данной секции вы можете видеть 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 домашнюю директорию. При значении `no` создание и изменение файлов станет невозможным;
- `create mode = 0750` — права доступа для создаваемых файлов. В данном случае владелец файла имеет полные права, пользователи группы могут читать и выполнять его, а остальные — бесправны. Иногда следует понизить значение параметра до 740, чтобы пользователи группы могли хотя бы читать;
- `directory mode = 0775` — права доступа для создаваемых каталогов. В данном случае у пользователей группы тоже слишком высокие права, и я бы понизил их до 755, чтобы запретить создавать в новой директории файлы. Остальные могут только просматривать каталог, но и это может оказаться лишним, и для большей безопасности лучшим решением будет значение 750;
- `valid users = пользователи` — указывается список пользователей, которым разрешен доступ к домашним директориям. 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 домен, 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 так же присутствует комментарий и директива `writable`. Пользователи не должны иметь право записи в эту директорию, потому что здесь хранятся их 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 директории **netlogon**;
- `guest ok = yes` — такую же директиву мы рассматривали в *разд. 6.1*. В данном случае она позволяет гостевой вход.

Помимо этого, нужно убрать комментарии в секции `[Profiles]`, которая выглядит следующим образом:

```
;[Profiles]
; path = /usr/local/samba/profiles
; browseable = no
; guest ok = yes
```

В этой директории хранятся профили пользователей, и она не должна быть видна в сетевом окружении Windows, поэтому директива `browseable = no` запрещает отображение.

6.2.3. Распечатка

Для того 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 "гости" смогли использовать принтер, то добавьте строку `public = yes`. Я не рекомендую этого делать, потому что это предоставит пользователям лишнюю возможность для подшучивания. Например, в моей сети был случай, когда сотрудник рассылал через принтер разные картинки на все компьютеры. Вроде безобидно, а работу тормозит, и идет бесполезный расход картриджа.

6.2.4. Общий доступ

Чаще всего, на сервере необходима директория, через которую любой пользователь сможет обмениваться файлами с другими участниками сети. Для настройки такой папки используется 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 директория открытая, и в нее могут записывать и 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 директорию могут писать любые пользователи, то злоумышленник может записать в нее свои файлы, которые помогут ему получить на сервере права администратора root.

6.2.5. Личные директории

Все разделы, которые мы рассматривали выше, имеют устоявшиеся имена и предназначены для решения определенных задач. Но в некоторых случаях вы можете изменить имя раздела, и на работе сервера это не скажется. Но я не рекомендую этого делать, потому что в одной версии Samba все будет работать хорошо, а при обновлении программ сервера все может измениться, и потом трудно будет найти ошибку, благодаря которой Samba работает неверно.

Но вы можете создавать собственные разделы, в которых будете описывать права. Например, вы можете захотеть организовать общую директорию, в которой файлы будут доступны всем пользователям, но записывать могут только члены определенной группы. Допустим, что в этом каталоге должны храниться какие-то картинки. Для решения этой 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 данного раздела задаются следующие директивы:

- `path = /home/samba/images` — директория, которую мы хотим открыть;
- `public = yes` — признак доступности;
- `writable = yes` — запись в каталог разрешена;
- `write list = @staff` — определяет группу `staff`, которая может записывать в файл. Для всех остальных директория открыта только для чтения.

В таких объявлениях директорий можно описывать любые директивы, которые мы рассматривали в данной главе.

6.3. Управление пользователями

Для начала разберемся с именами пользователей. Для доступа к серверу Samba используются сведения из системного файла **/etc/passwd**. Но вы можете завести отдельные записи Samba-сервера, которые будут соответствовать реальным именам, но их можно будет использовать только для подключения к Samba, а не к системе.

Имена Samba-пользователей описываются в файле **/etc/samba/smbusers**, расположение и название которого могут быть изменены через директиву `username map` файла smb.conf. Содержимое файла может быть таким:

```
# Unix_name = SMB_name1 SMB_name2
root = administrator admin
nobody = guest pcguest smbguest
```

У списка имен несколько предназначений. Например, с его помощью вы можете спроецировать имена, привычные для пользователей DOS и Windows, на учетные записи Linux. Например, в Windows максимальные права принадлежат пользователю Administrator, а в Linux — это root. Во второй строке приведенного выше примера устанавливается соответствие имени `administrator` пользователю `root`.

Второе назначение файла — аккумулировать несколько имен на одной учетной записи. Например, у вас есть группа пользователей, которым необходимо назначить одинаковые права. Для этого создаем в Linux только одну учетную запись `nobody` (под ней будут работать пользователи), а в smb заводим не-

сколько 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 через директиву `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 на несколько колонок. Наиболее интересные из них первые три — имя пользователя, его UID в Linux-системе и пароль.

Но добавлять пользователей вручную не очень удобно, потому что нужно зашифровать и прописать пароль, что не так уж и просто. Чтобы облегчить задачу, в пакет Samba включена утилита smbpasswd, которая имеет следующие параметры:

- `a` — добавить пользователя в Samba-систему. Учетная запись должна уже существовать в **/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. Я даже рекомендую сделать их различными. ОС Windows умеет запоминать пароли и хранить в своей системе, а версии Windows 9*x* делают это небезопасно. 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. Таким образом, вы легко можете определить, какие записи активны, а какие нет.

- `e` — активировать пользователя. С помощью этой команды можно подключить пользователя: `smbpasswd -d robert`.

Дополнительные параметры этой утилиты можно увидеть в файле помощи `man`.

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 `smdpasswd`. 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 доступ в систему. К таким пользователям относятся системные учетные записи типа bin, adm, deamon и др.

6.4. Использования Samba

Сервис Samba в основном создавался для пользователей Windows, но и поклонники Linux тоже оценили все преимущества этой технологии, тем более что Linux выполнила задачу разделения файлов по сети не хуже Windows, а где-то даже лучше. Для работы с Samba из ОС Linux используется команда `smbclient`.

Чтобы подключиться к серверу, необходимо, как минимум, указать 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, иначе 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 следующим образом:

```
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 данном списке находятся не все директории. Например, у домашних каталогов из раздела [homes] файла конфигурации директива browseable установлена в значение no *(см. разд. 6.2.1)*. Следовательно, таких каталогов не будет видно. Это вполне логично, потому что нельзя злоумышленнику давать возможность лицезреть имена директорий, 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 написать 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 домашнюю директорию пользователя flenov. Ее адрес будет **\\192.168.1.1\flenov**.

Но здесь надо сделать одно замечание — в Linux обратный слэш ("\") является служебным, поэтому каждый такой знак должен заменяться двумя символами "\\". Так как в начале UNC-имени идет "\\", то они подменяются на четыре обратных слэша, и адрес, приведенный выше, должен выглядеть как **\\\\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, то необходимо указать имя пользователя, обладающего правами:

```
smbclient \\\\192.168.1.1\\flenov -U flenov
```

При удачном подключении к открытому ресурсу вы увидите приглашение, в котором можно выполнять различные операции над файлами:

```
Smb: \>
```

Чтобы узнать, какие команды доступны, введите директиву `help` или знак вопроса "?". Команды, которые вы увидите, очень похожи на FTP (более подробно с FTP мы познакомимся в *гл. 10*). Для отключения от ресурса необходимо выполнить команду `exit`.

Большинство дистрибутивов Linux включают стандартный Samba-пакет, и ничего больше. А ведь в Интернете есть сторонние разработки, которые позволяют монтировать открытые ресурсы в файловую систему Linux как дискету или CD-ROM или работать с общими ресурсами в графическом режиме, как это делается в Windows.

ГЛАВА 7

Web-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 Интернет, но и интранет без Web-страниц.

Для того чтобы пользователь смог просматривать с вашего узла хотя бы простейшие Web-странички, необходим Web (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 больше половины. Хотя для Linux существуют и другие Web-серверы (например, TUX), но когда говорят о Web-сервере под Linux, то подразумевают именно Apache.

Apache абсолютно бесплатный сервер, который распространяется по лицензии GNU и доступен не только для Unix-систем, но и для ОС Windows. Официальный сайт разработчиков — **www.apache.org**. Почему этот сервер стал так популярен? Неужели так повлиял фактор дармовщины? Бесплатность, конечно, соблазнительна, но качество превыше всего. Сервер Apache располагает громадным количеством возможностей и при этом обладает следующими немаловажными особенностями:

- безопасность — многие профессионалы относят к разряду самых защищенных;
- надежность — в сочетании с ОС Linux может работать без перезагрузок годами;
- неприхотливый— работает с минимальной нагрузкой на систему (не требует особых ресурсов);
- производительность — высокая скорость отклика и обработки запросов пользователей.

На основе Apache строят серверы, которые работают практически без выключения (доступны 99,9 %). Многие корпорации вверяют ему свои важные данные, и мне не известны случаи, когда кто-нибудь пожалел о содеянном.

Единственный недостаток, который отмечают пользователи, — это сложность конфигурирования. Приходится редактировать текстовый файл, а т. к. настроечных параметров огромное количество, то это может оказаться утомительным занятием. При этом нет гарантии, что все настройки будут сделаны оптимальными и безопасными, ведь очень легко упустить что-либо.

В данной книге мы не сможем рассмотреть все возможные параметры, потому что их слишком много. Мы коснемся только основных и поговорим о том, что может сказаться на производительности и безопасности сервера.

7.1. Основные настройки

Основные настройки сервера Apache располагаются в файле **/etc/httpd/conf/httpd.conf** (для некоторых дистрибутивов — в **/etc/httpd.conf**). Здесь хранятся настройки Web-сервера, его виртуальных серверов и программных модулей. Для Red Hat Linux, а именно он взят за основу в этой книге, все рассматриваемые параметры сервера находятся в этом файле, если явно не указано другое расположение.

Как и для большинства других сервисов, в современных дистрибутивах для настройки Apache есть простая и удобная графическая утилита. Для ее запуска нужно в основном меню системы выбрать **System/Apache configuration**. На рис. 7.1 показано главное окно программы Apache Configuration, позволяющей быстро и легко настроить Web-сервер.

С помощью визуальной программы очень удобно делать первоначальные настройки, но после этого вы обязательно должны самостоятельно просмотреть конфигурационный файл, а для этого нужно знать его параметры.

Внимание!

Если вы вручную внесли исправления в конфигурационный файл, то после этого использовать графическую утилиту нельзя, потому что она может неправильно интерпретировать ваши правки и поменяет их на свои значения. Чтобы изменения вступили в силу, необходимо перезапустить сервер. Сервис Apache считывает параметры из конфигурационного файла только во время старта.

За счет прямого редактирования файла конфигурации можно добиться максимально безопасной и быстрой работы. Рассмотрим основные параметры Web-сервера:

- ServerType — тип сервера, может принимать значения inetd или standalone. Если выбрать inetd, то такие параметры, как порт, игнорируются, а используются значения из конфигурации демона inetd *(см. разд. 5.4)*;

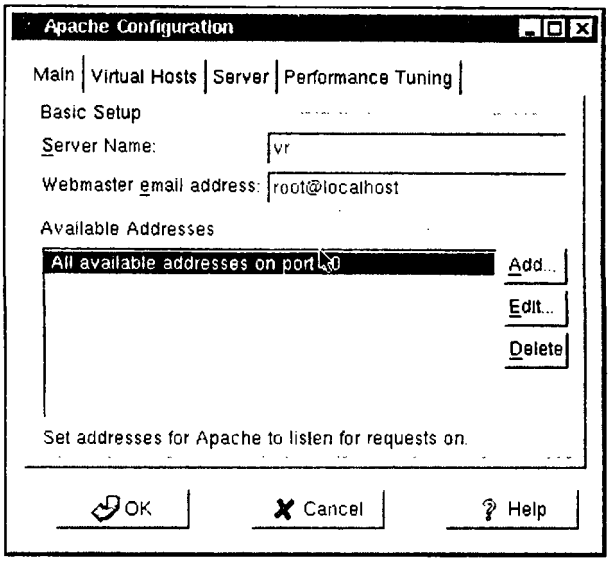

Рис. 7.1. Главное окно программы Apache Configuration

- `ServerRoot` — корневая директория, в которой находятся файлы конфигураций и журналы;
- `Timeout` — предельное время ожидания в секундах для получения и отправки пакетов;
- `Port` — порт, на котором будет работать сервис. По умолчанию и для общедоступных серверов используется значение `80`. Но для закрытых серверов, которыми пользуется узкий круг людей, можно изменить значение, например, на 10387. В этом случае в качестве адреса страницы надо использовать URL типа `ИмяСервера:10387` (например, `www.linux.com:10387 /index.htm`). Таким образом, не зная порта, злоумышленник не сможет проникнуть в систему, пока не просканирует весь диапазон портов и не выяснит, что 10387 — это порт Web-сервера. Это простейшая, но достаточно эффективная защита от скриптеров, которые обладают минимумом знаний о безопасности и взламывают компьютеры только с помощью сплоитов, написанных другими хакерами;
- `ServerTokens` — при обращении к серверу 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.

  Очень опытные администраторы могут даже подменить имя сервера, но для этого необходимо будет перекомпилировать исходные коды Apache. Заголовок хранится в файле **include/ap_releas.h** в виде следующих двух строк:

```
#define SERVER_BASEPRODUCT "Apache"
#define SERVER_BASEVERSION "2.0"
```

  Подставьте имя какого-либо другого сервера. Желательно, чтобы оно было похоже на реальность, иначе профессиональный хакер сразу заметит обман.

  В более ранних версиях Apache расположение файла было другим, но мы же договорились, что устанавливать надо все только последнее;

- HostnameLookup — если установлено значение on, то IP-адрес клиента, запросившего данные с сервера, будет преобразован в доменное имя, иначе будет использоваться IP-адрес;
- User и Group — группа и имя пользователя, с правами которого будет работать сервис. По умолчанию в Linux для этих параметров используется Apache. Этот пользователь и группа должны обладать минимальными правами в системе, которые только необходимы для работы Web-сервера и его модулей. Ничего лишнего разрешать им нельзя;
- ErrorLog и CustomLog — определяют местоположение журналов;
- LogLevel — обусловливает, что будет писаться в журнал. 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 расходу лишних ресурсов. Допустим, что пользователь запросил страницу с 10 картинками. В ответ на это браузер клиента

страницу с 10 картинками. В ответ на это браузер клиента откроет одиннадцать соединений с Web-сервером (одно для получения документа html и десять — для картинок документа). Если включить этот параметр, то за одно подключение может быть обработано несколько запросов;

- `MaxKeepAliveRequests` — максимальное число запросов, которое может быть выполнено в течение одного соединения;
- `KeepAliveTimeout` — время ожидания в секундах очередного запроса от одного и того же клиента. Если за указанный период запрос не поступит, то соединение завершается;
- `MaxClients` — максимальное количество клиентов, которые могут подключиться одновременно. Если сделать это значение слишком большим, то злоумышленник сможет произвести атаку DoS (отказ от обслуживания), открыв слишком много соединений, с которыми не справится сервер. По умолчанию установлено `150`, но этого будет достаточно только для небольшого сервера. Apache способен обработать намного больше даже на слабеньком железе. Вы должны указать такое значение, при котором сервер сможет обрабатывать максимальное число запросов и при этом успешно работать (не зависнуть) с предельной загрузкой;
- `MaxRequestsPerChild` — число запросов, которое позволено обрабатывать дочернему процессу до переполнения. Если Apache или используемые им библиотеки допускают некорректную работу с памятью (выделяют, но не освобождают) или другими ресурсами, то во избежание проблем при длительной непрерывной работе сервера дочерний процесс при переполнении будет принудительно завершен. Большинству систем это не требуется, но некоторые (например, Solaris) страдают заметными "утечками" в библиотеках.

7.2. Модули

При настройке сервиса Apache очень важным звеном являются модули. Загрузка их описана в конфигурационном файле **/etc/httpd/conf/httpd.conf** следующим образом:

```
<IfDefine HAVE_PERL>
LoadModule perl_module modules/libperl.so
</IfDefine>
```

В первой строке проверяется параметр `HAVE_PERL`. Если он установлен, то команда `LoadModule` загружает модуль **modules/libperl.so**, необходимый для интерпретации Perl-сценариев.

Следом идет подключение модулей, которое похоже на загрузку, но используется команда AddModule:

```
<IfDefine HAVE_PERL>
AddModule mod_perl.c
</IfDefine>
```

По умолчанию загружаются все установленные модули или те, которые включены в дистрибутив. Но это не оптимально, потому что разработчик не может знать, что нам понадобится. Рассмотрим основные модули поддержки сценариев, которые могут загружаться:

- perl_module — Perl;
- php_module — PHP;
- php3_module — PHP версии 3;
- php4_module — PHP версии 4;
- python_module — на языке Python.

Эти модули являются наиболее опасными для Web-сервера, потому что позволяют выполнять сценарии, через которые могут происходить взломы. Например, воспользовавшись ошибкой в сценарии PHP, злоумышленник может выполнять какие-либо команды на сервере. Хорошие сайты используют какой-то один язык Web-программирования: Perl, PHP или Python, и вы должны оставить только тот модуль, который необходим, а остальные лучше отключить.

Я рекомендую взять на вооружение для программирования PHP, потому что он гибок в настройках и может обеспечить большую безопасность. Да и по моему опыту, злоумышленники чаще используют Perl для создания rootkit (набор администратора или программа, которая позволяет выполнять необходимые хакеру действия на удаленной системе) на взломанной системе. Но это мое мнение, которое отнюдь не претендует на истину в последней инстанции. Хороший программист на Perl сможет легко написать программу, которая будет абсолютно безопасной и доставит много хлопот хакеру. На любом языке, даже самом "дырявом", можно написать супер защищенную программу, а на самом проверенном — наделать дыр. Это уже полностью зависит от программиста и его умений.

Неиспользуемые модули необходимо отключить, это значительно сузит возможности злоумышленника. Ничего не напоминает вам такая рекомендация? Да, это снова самое популярное мое правило: лишняя программа — враг администратора и дополнительная дверь для хакера.

Обязательно уделите внимание рассмотрению загружаемых модулей и удалите или закомментируйте то, что ненужно. Сделав это, вы повысите

безопасность Web-сервера более чем на 50 %. Почему? Если Python хакерами используется редко, то Perl и PHP очень популярны. Получается, что у вас в системе две большие двери. Если закрыть хотя бы одну, то останется ровно половина.

7.3. Права доступа

В данном разделе нам предстоит познакомиться с основными параметрами файла конфигурации **/etc/httpd/conf/httpd.conf**. Они отвечают за права доступа, относящиеся к директориям, и имеют следующий вид:

```
<Directory /var/www/html>
  Order allow,deny
 Allow from all
</Directory>
```

или, например:

```
<Location /server-status>
 SetHandler server-status
 Order deny,allow
 Deny from all
 Allow from .your-domain.com
</Location>
```

Первое объявление задает права доступа к определенной директории на диске (в данном случае **/var/www/html**), а второе — ограничивает доступ к виртуальной директории (в приведенном примере **http://servername/server-status**).

Если вы знакомы с HTML (HyperText Markup Language, язык разметки гипертекста) или XML (Extensible Markup Language, расширенный язык разметки), то такое объявление будет вам известно и понятно без дополнительных разъяснений. Для тех, кто не в курсе, я сделаю несколько пояснений на примере директорий. 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 директории могут быть описаны не только в файле **/etc/httpd/conf/httpd.conf**, но и в файле **.htaccess**, который может находиться в указанном каталоге. Сам файл требует отдельного рассмотрения *(см.*

разд. 7.5.1), а сейчас вы должны только знать, что разрешения, указанные в конфигурации Web-сервера, могут быть переопределены.

Теперь рассмотрим, как задаются права доступа. Для этого используются следующие директивы:

- `Allow from параметр` — определяет, с каких хостов можно получить доступ к указанной директории. В качестве `параметр` можно использовать одно из следующих значений:
  - `all` — доступ к директории разрешен всем;
  - `доменное имя` — определяет домен, с которого можно получить доступ к директории. Например, можно указать domain.com. В этом случае только пользователи этого домена смогут получить доступ к директории через Web. 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 директории 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 директории определенной сетью;
  - `env=ИмяПеременной` — доступ разрешен, если определена указанная переменная окружения. Полный формат директивы: `allow from env=ИмяПеременной`;
- `Deny from параметр` — запрещение доступа к указанной директории. 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`. Может быть три варианта:
  - `Order deny, allow` — изначально все разрешено, потом первыми применяются запреты, а затем разрешения. Рекомендуется использовать только на общих директориях, в которые пользователи могут самостоятельно закачивать файлы, например, свои изображения;
  - `Order allow, deny` — сначала все запрещено, вслед за этим применяются разрешения, затем запрещения. Рекомендуется применять для всех директорий со сценариями;
  - `Order mutual-failure` — исходно запрещен доступ всем, кроме перечисленных в `allow` и в то же время отсутствующих в `deny`. Советую ис-

пользовать 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 директории разрешен любому пользователю, прошедшему аутентификацию;
- `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`;
- `AllowOverwrite параметр` — определяет, какие директивы из файла **.htaccess** в указанном каталоге могут перекрывать конфигурацию сервера. В качестве `параметр` 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 [+ | -] параметр` — определяет возможности Web-сервера, которые доступны в данном каталоге. Если у вас есть директория, в которую пользователям разрешено закачивать картинки, то вполне логичным является запрет на выполнение в ней любых сценариев. Не надо надеяться, что вы сумеете программно запретить загрузку любых файлов, кроме изображений. Хакер может найти способ закачать злостный код и выполнить его в системе. С помощью опций можно сделать так, чтобы сценарий не выполнился Web-сервером.

  Итак, после ключевого слова можно ставить знаки плюс или минус, что соответствует разрешению или запрещению опции. В качестве `параметр` указывается одно из следующих значений:
  - `All` — все, кроме `MultiView`. Если указать строку `Option + All`, то в данном каталоге будут разрешены любые действия, кроме `MultiView`, который задается отдельно;
  - `ExecCGI` — разрешено выполнение CGI-сценариев. Чаще всего для этого используется отдельная директория **/cgi-bin**, но и в ней можно определить отдельные папки, в которых запрещено выполнение;
  - `FollowSymLinks` — позволяет использовать символьные ссылки. Убедитесь, что в директории нет опасных ссылок и их права не избыточны.

Мы уже говорили в *разд. 3.1.3* о том, что ссылки сами по себе опасны, поэтому с ними нужно обращаться аккуратно, где бы они ни были;

- `SymLinksIfOwnerMatch` — следовать по символьным ссылкам, только если владельцы целевого файла и ссылки совпадают. При использовании символьных ссылок в данной директории лучше указать этот параметр вместо `FollowSymLinks`. Если хакер сможет создать ссылку на каталог **/etc** и проследует по ней из Web-браузера, то это вызовет серьезные проблемы в безопасности;
- `Includes` — использовать SSI (Server Side Include, подключение на сервере);
- `IncludesNOEXEC` — использовать SSI, кроме `exec` и `include`. Если вы не используете в сценариях CGI эти команды, то данная опция является 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 как **www.cydsoft.com/index.htm**. В первом варианте сервер сам ищет файл по умолчанию и открывает его. Это могут быть index.htm, index.html, index.asp или index.php, default.htm и т. д. Если один из таких файлов по указанному пути не найден, то при включенной опции `Indexes` будет выведено дерево каталога, 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.

Все выше описанные директивы могут использоваться не только в файле **/etc/httpd/conf/httpd.conf**, но и в файлах **.htaccess**, которые могут располагаться в отдельных директориях и определять права этой директории.

Права доступа могут назначаться не только на директории, но и на отдельные файлы. Это описание делается между двумя следующими строками:

```
<Files ИмяФайла>
</Files>
```

Это объявление может находиться внутри объявления прав доступа к директории, 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 такие же, как и для директорий. Исходя из этого, в данном примере к подкаталогу **/var/www/html** разрешен доступ всем пользователям, а к файлу **/var/www/html/admin.php** из этой директории запрещен доступ абсолютно всем.

Помимо файлов и директорий можно ограничивать и методы HTTP-протокола, 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 GET-метод, то запретите все остальные, чтобы хакер не смог воспользоваться потенциальной уязвимостью в сценарии, изменив метод.

Бывают варианты, когда не всем пользователям должно быть позволено отправлять данные на сервер. Например, сценарии в определенной директории 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, этот процесс схож с описанием разрешений на файлы. Даже права доступа используются те же самые, и размещаются внутри определения директорий (`<Directory >` или `< Location >`), и влияют только на указанный каталог.

К примеру, так можно запретить любую передачу данных на сервер в директории **/home**:

```
<Directory /home>
<Limit GET POST>
 Deny from all
</Limit>
</Directory>
```

Внутри определения прав для директории **/home** устанавливается запрет на методы GET и POST.

Ваша задача как администратора настроить такие параметры доступа на директории и файлы, при которых они будут минимально достаточными. Поль-

зователь не должен иметь возможности сделать ни одного лишнего шага. Для реализации этого вы должны действовать по правилу "Что не разрешено, то запрещено".

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 разрешение, которое позволит хакеру уничтожить мой сервер.

7.4. Создание виртуальных Web-серверов

На одном физическом сервере может работать большое количество виртуальных Web-серверов, например, **www.your_name.com** и **www.your_company.com**. Это два разных Web-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 (с внедрением протокола IPv6 эта проблема будет стоять не так остро). Виртуальные Web-серверы могут иметь как отдельные IP-адреса, так и использовать общий адрес, а различаться будут доменными именами;
- упрощение администрирования и контроля за безопасностью. Конфигурация Web-сервера и его защита — достаточно сложный процесс, поэтому намного легче настроить и обновлять программное обеспечение одного физического сервера, чем сотни серверов, ресурсы которых используются на 10 %.

Для создания виртуального сервера используется 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` — директория, в которой расположен корневой каталог сайта;
- `ServerName` — имя сервера. 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 директорию **/var/www/your_server/**, которая является корнем для виртуального Web-сервера. Разрешения можно устанавливать как внутри объявления виртуального сервера, так и вне его.

За более подробной информацией обратитесь к документации по Apache.

7.5. Замечания по безопасности

В конфигурационном файле **/etc/httpd/conf/httpd.conf** 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 директива 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 директориям. Например, если у вас есть каталог,

работа с которым разрешена только авторизованными пользователями, то можно указать файл паролей, в соответствии с которым доступ к файлам будет запрещен самим сервером.

7.5.1. Файлы .htaccess

Если какая-либо директория Web-сервера должна иметь особые права, то лучше создать в этом каталоге файл **.htaccess**. Разрешения, описанные в таком файле, действуют на директорию, в которой он находится. Рассмотрим пример содержимого файла **.htaccess**:

```
AuthType Basic
AuthName "By Invitation Only"
AuthUserFile /pub/home/flenov/passwd
Require valid-user
```

В данном файле для текущей директории 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. 7.2 приведен пример такого окна.

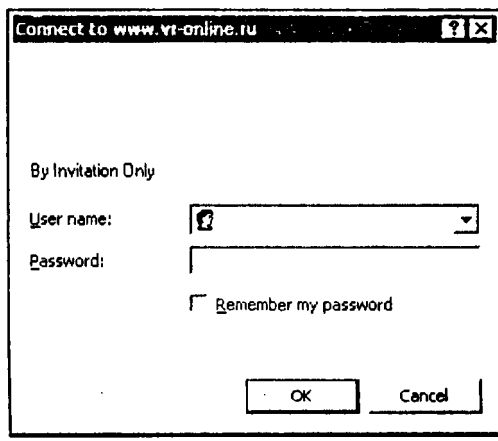

Рис. 7.2. Окно запроса имени и пароля

Директива `AuthUserFile` задает файл, в котором находится база имен и паролей пользователей сайта. Последняя команда `Require` в качестве параметра использует значение `valid-user`. Это значит, что файлы в текущей директории смогут открыть только те пользователи, которые прошли аутентификацию.

Вот таким простым способом можно запретить неавторизованный доступ к директории, содержащей секретные данные или сценарии администратора.

Как я уже говорил, в файле **.htaccess** могут находиться и директивы типа `allow from`, которые мы рассматривали выше *(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 хакера по взлому сервера сильно усложнится. Здесь уже недостаточно знания пароля, необходимо иметь конкретный IP-адрес для обращения к содержимому директории, а это требует значительных усилий.

Эти же параметры можно указывать и в файле httpd.conf, например:

```
<directory /путь>
AuthType Basic
AuthName "By Invitation Only"
AuthUserFile /pub/home/flenov/passwd
Require valid-user
</directory>
```

Чем будете пользоваться вы, зависит от личных предпочтений. Мне больше нравится работать с файлом **.htaccess**, потому что настройки безопасности будут храниться в той же директории, на которую устанавливаются права. Но это небезопасно, потому что хакер может получить возможность прочитать этот файл, а это лишнее.

Использование централизованного файла httpd.conf дает преимущества, т. к. он находится в директории **/etc**, которая не входит в корень Web-сервера и должна быть запрещена 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}lZZEBtPy4/gdHsyztjUEWb0d90E=
```

В этой записи два параметра, разделенных двоеточием. Сначала указано имя пользователя, а после разделителя — зашифрованный по алгоритму MD5 пароль. Формировать такой файл вручную сложно и не имеет смысла, поэтому для облегчения жизни администраторов есть утилита htpasswd. С помощью этой программы создаются и обновляются имена и пароли для базовой аутентификации Web-сервером HTTP-пользователей.

Удобство программы состоит в том, что она может шифровать пароли и по алгоритму MD5, и с помощью системной функции `crypt()`. В одном файле могут находиться записи с паролями обоих типов.

Если вы храните имена и пароли в формате базы данных DBM (для указания этого в файле **.htaccess** используется директива `AuthDBMUserFile`), то для управления БД нужно применять команду `dbmmanage`.

Давайте рассмотрим, как пользоваться программой htpasswd. Общий вид вызова команды выглядит следующим образом:

```
htpasswd параметры файл имя пароль
```

Пароль и имя файла являются необязательными, и их наличие зависит от указанных опций. Давайте посмотрим основные ключи, которые нам доступны:

- `-c` — создать новый файл. Если указанный файл уже существует, то он перезаписывается, и старое содержимое теряется. Пример использования команды:

  ```
  htpasswd -c .htaccess robert
  ```

  После выполнения этой директивы перед вами появится приглашение ввести пароль для нового пользователя robert и подтвердить его. В результате будет создан файл **.htaccess**, в котором будет одна 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 Web-сервер Apache. Такой ключ удобен, если у вас разнородная сеть, и один файл с паролями используется на разных серверах;

- `-d` — для шифрования будет применяться системная функция `crypt()`;

- `-s` — применить SHA-шифрование (на базе алгоритма хэширования), которое используется на платформе Netscape;

- `-p` — не шифровать пароли. Я не рекомендую устанавливать этот флаг, 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
```

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 зна-

комых такой длинный пароль пока еще никто не захотел устанавливать, а задавать 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 декодирует практически моментально.

Для создания реально безопасного соединения необходимо сначала зашифровать весь трафик. Для этого может использоваться stunnel или уже готовый протокол HTTPS, который использует SSL. О нем мы еще поговорим в *разд. 10.9*.

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, используется 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 не установлен интерпретатор языка Perl, то первую строку следует закомментировать, чтобы она даже не смущала. Вторая строка безобидна, а вот если таким же образом разрешить серверу работать с HTM- или HTML-файлами, то это уже станет небезопасным. Следующей строки не должно быть в вашем конфигурационном файле:

```
AddHandler server-parsed .html
```

Если где-то действительно есть необходимость подключения HTML-документа, то пропишите это в файле **.htaccess**. В остальных директориях я рекомендую в явном виде запретить обработку HTML-файлов сервером. Для этого добавьте следующую строку в конфигурационный файл httpd.conf или в файл **.htaccess** каждой директории:

```
RemoveHandler .html .htm
```

Таким образом, мы запретим выполнение файла на сервере, но не отменим SSI-инструкции. Например, следующий код в SHTML-файле будет выполнен:

```
<!--#include virtual="filename.shtml" -->
```

Если вы не используете SSI и, соответственно, SHTML-файлы, то закомментируйте следующую строку (по умолчанию она доступна):

```
AddHandler server-parsed .shtml
```

7.6. Проще, удобнее, быстрее

Процесс конфигурирования должен быть максимально удобным. Если все настройки будут нагромождены в одном файле **/etc/httpd/conf/httpd.conf**, то разобраться в них станет очень сложно. А чем больше параметров, тем выше вероятность, что вы что-либо прозеваете. Чтобы упростить поддержку вашего Web-сервера, могу посоветовать придерживаться следующих рекомендаций:

- для удобства администрирования все описания прав доступа можно перенести в конфигурационный файл **/etc/httpd/conf/access.conf**. По умолчанию этот файл пустой, а все используют только **/etc/httpd/conf/httpd.conf**. Выделение части разрешений в отдельный файл действительно может помочь в управлении, потому что права будут видны, как на ладони;
- основные настройки сервера, которые редко изменяются, можно перенести в конфигурационный файл **/etc/httpd/conf/access.conf**;
- комментируйте все ваши действия. Многие настройки не изменяются годами, и уже через пару месяцев трудно вспомнить, зачем вы установили определенную директиву. Например, вы запретили доступ для всех пользователей к какой-либо директории, которая временно использовалась для тестирования сценариев. Через какое-то время вы можете забыть это и случайно откроете доступ к сценариям, которые отлажены не полностью и могут стать причиной взлома и крушения системы.

Чем удобнее управлять безопасностью сервера, тем меньше ошибок вы совершите, потому что все параметры правильно сгруппированы, и подробные комментарии напоминают вам о назначении сделанных настроек. Такой подход к администрированию также помогает 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 Web-сервера приемлемо только на небольших сайтах. Когда работает сотня виртуальных серверов, то такие описания становятся слишком громоздкими. Даже

если все права выделить в отдельный файл **/etc/httpd/conf/access.conf**, его размер будет слишком велик, чтобы найти в нем что-то нужное.

Для больших сайтов я рекомендую описывать в конфигурационных файлах сервера только общие правила, которые затрагивают сразу несколько директорий. Это возможно, потому что при указании пути к каталогу можно использовать регулярные выражения. Приведу пример, который определяет правила для всего, что находится в директории **/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 директории значение `/home/*/public_html`, то все каталоги **public_html** в директории **/home** будут иметь указанные права, если нет явного переопределения для отдельных папок.

7.7. Безопасность сценариев

Как мы уже говорили, сценарии являются очень опасными для Web-сервера *(см. разд. 1.1.2)*. Через их ошибки происходило очень много громких взломов. Мы уже знаем, что необходимо отключить все интерпретаторы, которые не используются вами, и оставить только то, что нужно. Таким образом 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.). Если вам нужен какой-то интерпретатор, то необходимо максимально ограничить его возможности.

Допустим, что ваш сайт использует сценарии PHP, и в нем есть функции, которые обращаются к системе, и если вы их неверно используете (например, не проверяются параметры, заданные пользователем), то злоумышленник имеет 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 мы не должны надеяться на их профессионализм, потому что все мы — люди, и нам свойственно ошибаться. Мы должны сделать все, чтобы погрешности 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. Многие администраторы отключают safe_mode. Это не совсем верно. Я всегда сначала проверяю, можно ли переписать сценарий, и если это нереально, то только тогда выключаю безопасный режим.

При настройке интерпретатора вы опять же должны действовать от запрета. Изначально следует закрыть все, что нужно и ненужно. А потом включать необходимые вашим сценариям опции. Лучше всего, если вы будете заниматься конфигурированием не только рабочего сервера, но и сервера, который используют программисты для разработки и отладки своих сценариев. В этом случае можно будет контролировать все установленные параметры.

Действия администратора должны быть тесно связаны с работой программиста сценариев для Web-сайта. Если разработчику нужны какие-то опции, то именно вы должны их включать на обоих серверах. О любых корректировках скриптов, влекущих за собой уменьшение (увеличение) прав доступа, разработчик должен вам сообщать, и настройки должны быть изменены.

Администратор и разработчик должны находиться в постоянном контакте, чтобы оперативно реагировать на необходимость использования каких-либо дополнительных возможностей. Некоторые администраторы избавляются от настройки интерпретаторов и переводят эти функции на разработчиков. Это не совсем правильно, потому что программист пишет код и не может достаточно хорошо разбираться в вопросах конфигурации серверов, чтобы обеспечить требуемый уровень безопасности.

Все настройки для интерпретатора PHP хранятся в файле **/etc/php.ini**. Мы этот файл не будем рассматривать, потому что эта тема выходит за рамки книги по Linux.

7.7.1. Основы безопасности

В настоящее время большинство взломов в Интернете совершается с помощью или благодаря ошибкам в сценариях Web-страниц. Давайте попробуем разобраться, почему это происходит.

Большинство владельцев домашних сайтов — это простые пользователи, которые хотят быстро получить собственную страницу с множеством возмож-

ностей. 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. Пользователи не хотят (или не могут) вникать в тонкости программирования, поэтому используют в своих проектах уже готовые (платные или бесплатные) движки.

Как я уже говорил, любая разработка содержит ошибки, просто о них до поры до времени не известно. А распространенная программа становится лакомым кусочком для любого хакера, потому что ее взлом позволяет приникнуть в систему, на которой установлена эта программа.

Если администратор сайта устанавливает на свою страничку популярный форум, то он должен понимать, что когда-нибудь в нем найдут уязвимость, и через нее любой злоумышленник проникнет в систему. Чтобы этого не произошло, администратор сайта должен регулярно обновлять используемые Web-программы и Web-сценарии.

Если вы решили написать собственный форум для сайта, то он может оказаться более надежным, если знать хотя бы основные принципы защиты Web-приложений. В этом случае ваша работа будет безопаснее, чем при использовании любой готовой программы и, тем более, программы с открытым кодом.

Ну а если вы не знаете особенностей языка или вообще не знакомы с программированием, то лучше даже не пытаться. В этом случае даже начинающий хакер найдет уязвимость, не зная исходного кода, структуры базы данных и других параметров, упрощающих взлом Web-сценария.

Как видите, сложности есть всегда. Самый безопасный вариант — использовать на Web-сайтах наименее распространенные программы, разработанные профессиональными программистами. Лучше всего, если они будут с закрытым кодом или даже написаны на заказ. Это требует дополнительных затрат, но расходы на восстановление системы после взлома намного больше.

Если вы отвечаете за один Web-сервер, то легко сможете контролировать обновление программ. Хуже всего администраторам хостинговых компаний. На их серверах расположены сотни, а то и тысячи Web-сайтов. Проследить за всеми хозяевами сайтов нереально, поэтому нужно защитить свои владения от недобросовестных или ленивых пользователей. Для этих целей лучше всего подходит программа jail, о которой мы говорили в *гл. 4*. Для Web-сервиса вы должны создать его собственный виртуальный сервер, в котором он и будет работать. Если злоумышленник взломает систему через Web-программу, то сможет нарушить работу только виртуальной директории.

Во время подготовки материалов для данной книги в одном из популярных форумов была найдена уязвимость, позволяющая выполнять любые команды

на удаленной системе. Для этого нужно было директиву специальным образом передать через строку URL. Эту главу я начал писать через месяц после этого страшного открытия, и почему-то вспомнив про ошибку, решил проверить серверы в Интернете. Я запустил поиск всех сайтов, содержащих уязвимый форум. 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 всю директорию **/etc**, и моих прав хватало даже на удаление файлов. Конечно же, делать этого я не стал даже в тестовых целях. Для доказательства прав я переименовал по одному файлу на каждой системе и сообщил об уязвимости 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 для этого отправляется анонимное письмо.

Конечно же, 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 (каждый работает в своем пространстве), выделенные серверы для каждого сайта, запрет на выполнение опасных функций в интерпретаторе PHP и т. д.

Выбрать какой-либо определенный метод защиты множества виртуальных серверов достаточно сложно, а иногда просто невозможно. Например, один сайт требует для своей работы интерпретатор Perl, а другой — PHP. Приходится включать обе возможности.

Из собственного опыта могу посоветовать использовать несколько физических серверов для разделения хранимых сайтов в соответствии с их потребностями:

- используется интерпретатор PHP в защищенном режиме;
- нужен интерпретатор PHP с полными правами;
- применяется интерпретатор Perl.

И так далее. Вы должны сгруппировать сайты, исходя из их требований, тогда администрирование станет намного удобнее и проще.

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 при этом пользователи не обновляют эти программы. Когда-нибудь злоумышленник взломает домашнюю страничку через уязвимые сценарии и сможет украсть номера кредитных карт пользователей интернет-магазина. Это поставит крест на вашей карьере администратора.

7.7.2. mod_security

Несмотря на то, что безопасность Web-сервера, в основном, зависит от выполняемых на нем сценариев и программистов, которые пишут эти скрипты, есть возможность защитить сервер вне зависимости от этих факторов. Отличное решение данной проблемы — бесплатный модуль для Apache под названием mod_security.

Принцип действия 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 Web-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.

Итак, модуль mod_security находится на сайте **http://www.modsecurity.org**. После его установки в файле httpd.conf можно будет использовать дополнительные директивы, фильтрации запросов. Рассмотрим наиболее интересные из них:

- `SecFilterEngine On` — включить режим фильтрации запросов;
- `SecFilterCheckURLEncoding On` — проверять кодировку URL (URL encoding is valid);
- `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 применить символ "конец строки", необходимо указать в URL-адресе "%13". В данном случае коды

символов менее 32 и более 126 являются недопустимыми для адреса, поэтому вполне логично такие пакеты не пропускать к Web-серверу;

- `SecAuditLog logs/audit_log` — определяет файл журнала, в котором будет сохраняться информация об аудите;
- `SecFilterDefaultAction "deny,log,status:406"` — задает действие по умолчанию. В данном случае 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/password` — устанавливает запрет на использование в запросе пользователя обращения к файлу **/etc/passwd**. Таким же образом нужно добавить ограничение на файл **/etc/shadow**;
- `SecFilter /bin/ls` — отказ пользователям в обращении к директивам. В данном случае запрещается команда `ls`, которая может позволить хакеру увидеть содержимое каталогов, если в сценарии есть ошибка. Необходимо предотвратить обращение к таким командам, как `cat`, `rm`, `cp`, `ftp` и др.;
- `SecFilter "\.\./"` — классическая атака, когда в URL указываются символы точек. Их не должно там быть;
- `SecFilter "delete[[:space:]]+from"` — запрет текста `delete ... from`, который чаще всего используется в SQL-запросах для удаления данных. Такая строка очень часто используется 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"` — позволяют защититься от XSS-атак (Cross-Site Scripting, межсайтовое выполнение сценариев).

Мы рассмотрели основные методы, использование которых может повысить безопасность вашего Web-сервера. Таким образом, можно защищать даже сети из серверов. Дополнительную информацию можно получить с сайта разработчиков.

7.7.3. Секреты и советы

Как бы аккуратно программист не писал сценарии, как бы вы их не защищали с помощью специальных модулей, неплохо предпринять дополнительные меры безопасности. Есть еще ряд параметров, которыми можно воспользоваться для этих целей. В данном разделе я собрал небольшие рекомендации, которые вам помогут сделать необходимые настройки.

Ограничение сценариев

Во-первых, ограничьте выполнение сценариев только отдельной директорией. Чаще всего это каталог **cgi-bin**. Однажды я видел систему, в которой для этих целей администратор указал корневой каталог, что позволяло хранить сценарии, где угодно. Не стоит повторять эту ошибку, потому что в системе могут быть различные программы, написанные на Perl, и они должны быть недоступны для выполнения на Web-сервере.

Резервные копии

Никогда не сохраняйте резервные копии сценариев в каталогах, доступных Web-серверам. Рассмотрим пример. 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, т. к. сервер Apache не передает таких данных клиенту.

Перед тем как вносить изменения в сценарий, программисты любят создавать на сервере 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, и если хакер запросит файл, то он увидит его исходный код, что поможет быстрее найти ошибки в скриптах.

Когда злоумышленник исследует сценарии на сервере, то никто не мешает проверить наличие резервной копии. Допустим, что у вас есть файл **www.servername.com/index.php**, хакер попробует загрузить **www.servername.com/index.bak** или **www.servername.com/index.old**. На любительских сайтах такие версии встречаются очень часто. Не допускайте подобных промахов.

Любой специалист по безопасности должен запретить пользователям работу с резервными копиями. Сколько бы мы не 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 безопасными, т. е. запретить доступ со стороны Web-клиентов.

Это можно сделать, используя следующие директивы:

```
<FilesMatch "\.bak$">
Order deny, allow
Deny from all
</FilesMatch>

<FilesMatch "\.old$">
Order deny, allow
Deny from all
</FilesMatch>
```

7.8. Индексация Web-страниц

За последние 10 лет Интернет разросся до таких размеров, что найти в нем что-либо без хорошей поисковой системы стало невозможным. Первые системы просто индексировали страницы по их содержимому и потом использовали полученную базу данных для поиска, который давал 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 по ним поиск 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 **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 функции и используют в своих целях.

Один из самых простых способов взлома — найти с помощью поисковой системы закрытую Web-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. В таких случаях Google проиндексирует запрещенные страницы, и их можно будет 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.com** можно найти достаточно важные данные, которые скрыты от пользователя, но по ошибке администратора стали доступными для индексирующей машины Google. Во время поиска нужно правильно за-

давать параметры. Например, можно ввести в строку поиска следующую команду:

```
Годовой отчет filetype:doc
```

Или

```
Годовой отчет filetype:xls
```

И вы найдете все файлы в формате Word и Excel, содержащие слова "Годовой отчет". Возможно, документов будет 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.

Давайте рассмотрим, как можно запретить индексацию каталогов Web-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 попадается под руку: текст, описания, названия картинок, документы поддерживаемых форматов (PDF, XLS, DOC 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 простой 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 вначале 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/
```

Данные 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 директории с секретными данными, то следует запретить их индексацию. Лучше лишний раз отказать, чем потерять. При этом не стоит слишком увлекаться и закрывать все подряд, потому что если сайт не будет проиндексирован, то его не найдут поисковые машины, и вы потеряете большое количество посетителей. Если поинтересоваться статистикой, то можно увидеть, что на некоторых сайтах количество посетителей, пришедших с поисковых систем, превышает заходы по любым другим ссылкам или входы напрямую.

7.9. Безопасность подключения

В *разд. 14.5* мы будем рассматривать различные технологии прослушивания сетевого трафика. В основном они эффективны в локальных сетях, но хакеры больше любят интернет-соединения, потому что здесь можно найти больше интересного и есть лазейка, чтобы удаленно проводить атаку.

Как можно, находясь в Европе, перехватить трафик, который проходит между двумя городами в США? Я думаю, что пакеты будут следовать по каналам США, и в Европе им делать нечего. Но задача хакера сделать свой компьютер посредником в передаче пакетов данных, что-то наподобие прокси-сервера.

Самое сложное — организовать, чтобы клиент подключился не к реальному Web-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 Web-серверу и возвращать ответы клиенту (рис. 7.3). Таким образом, весь трафик будет проходить через компьютер хакера.

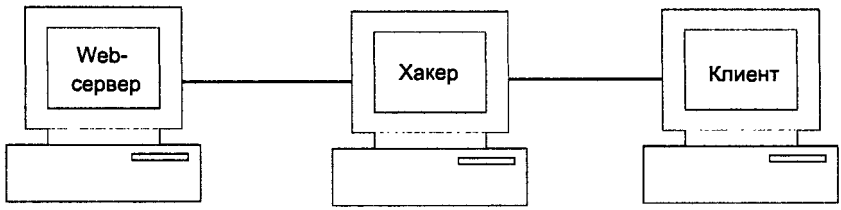

Рис. 7.3. Перехват трафика

Что опасного может увидеть хакер, когда клиент просматривает страницы на Web-сервере? Мы каждый день вводим на Web-страницах какие-либо данные, пароли, номера кредитных карт, и именно это является основной целью хакера. Но этот метод был хорош несколько лет назад, когда не было HTTPS-протокола и безопасного соединения с помощью SSL-шифрования.

Давайте вспомним, что для подключения по SSL любая программа-клиент (например, браузер) и Web-сервер обмениваются ключами, с помощью которых происходит шифрование. Для HTTPS помимо открытого и закрытого ключей необходимы подтвержденные сертификаты, которые выдаются специализированными компаниями. 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 просто будет передавать данные между клиентом и сервером, то трафик останется зашифрованным, и его просмотреть 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 Web-сервером и получает его ключ.
5. Между компьютером хакера и Web-сервером устанавливается соединение по ключу Web-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 са-

мое страшное — большинство пользователей, которые давно работают с Интернетом, устали смотреть на различные предупреждения, поэтому, не читая сообщения, нажимают кнопку **OK** для продолжения работы.

Решить проблему подложного подключения можно только защитой DNS-сервера, чтобы хакер не смог стать посредником в соединении между клиентом и сервером. Не используйте прокси-сервер, в происхождении которого вы не уверены, ведь он может принадлежать хакеру, и тогда весь ваш Web-трафик оказывается в опасности.

Можно еще попытаться перевоспитать пользователей, чтобы они читали все сообщения, которые отображает браузер. Но это сложно. Для этого необходимо, чтобы Web-браузер графически (иконками) ранжировал информацию по степени критичности. Таким образом, увидев сообщение об опасности, пользователь обязательно его прочтет. Если оповещение о подключении к сайту с неподписанным сертификатом сделать важным, то пользователи испугаются и прервут соединение. А ведь таких сайтов достаточно, и многие из них вполне уважаемые и защищенные. Просто подпись стоит денег, а не каждый хочет вкладывать дополнительные средства. А подписанными являются только коммерческие проекты. Но даже среди таких авторитетных ресурсов бывают проблемы. Сертификат действителен только в течение указанного в нем периода, а если администратор не уследил за датой, то он устареет.

Единственное, что должен помнить пользователь, при неавторизованном соединении нельзя вводить номера кредитных карт. Вот это предупреждение должно появляться большими буквами при подключении к серверу с неподписанным сертификатом.

ГЛАВА 8

Электронная почта

Для кого-то Интернет — это просмотр сомнительных страниц на WWW, для некоторых — способ найти соперника 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 (чаты, IRC, ICQ и т. д.), электронная почта жила, живет и будет жить. Именно с электронной почты начали развиваться сети, и это был один из первых сервисов Интернета.

Лично для меня почтовый клиент стал основной программой, которой я пользуюсь чаще всего. Это переписка с читателями, друзьями, начальством и т. д. Так уж получилось, что люди, с которыми я работаю, находятся в других городах и даже странах. Расстояние до самых ближайших партнеров более 1000 км, а до издательства — 1600 км. Если раньше это было сопряжено с большими проблемами, то теперь, благодаря компьютеру, я могу жить в теплых краях, а выполнять работы для компании, которая находится на Аляске. Таким образом, легко сохранить ноги в тепле, а зарабатывать деньги в холодных областях.

Как работает электронная почта (E-mail)? Рассмотрим основные моменты отправки письма:

1. Пользователь создает в почтовом клиенте (программа 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 — SMTP-протокол.
2. Сервер, получив письмо, определяет место назначения. 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. С помощью DNS можно узнать IP-адрес сервера **servername.com**, которому должно быть доставлено письмо.

3. Письмо направляется серверу, на котором зарегистрирован получатель.
4. Получив письмо, сервер **servername.com** помещает его в почтовый ящик пользователя djon.
5. Пользователь просматривает свой почтовый ящик с помощью почтового клиента и может скачать письмо для чтения.

Описанный процесс похож на работу традиционной почты. Вместо серверов там выступают почтовые отделения, которые сортируют почту в зависимости от адреса назначения и передают письма на узел связи получателя.

Как я уже заметил, для передачи сообщений используется протокол SMTP, разработанный еще на заре становления Интернета. Его функций уже давно недостаточно, но он не утратил своей актуальности.

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 должного распространения и в настоящее время практически не используется.

Для приема почты есть три протокола:

1. POP3 — Post Office Protocol v3 (Почтовый протокол), в настоящее время наиболее распространенный протокол приема почты.
2. 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.
3. MAPI — Messaging Application Programming Interface (Интерфейс прикладного программирования электронной почты корпорации Microsoft), используется в сетях Microsoft на серверах Microsoft Exchange.

Самым распространенным средством доставки почты в Linux является самая старая программа sendmail. Она обладает большими возможностями, но достаточно сложна в использовании. Из-за своего почтенного возраста в сервере sendmail сохранилась возможность передачи по протоколу UUCP, что сейчас не так часто встретишь.

Принцип же работы sendmail достаточно прост. Получив письмо от клиента, программа определяет получателя и заносит необходимую для доставки служебную информацию в заголовок этого 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**. Сервер sendmail имеет плохую репутацию в связи со сложностью настройки. Действительно, если даже бегло посмотреть на **/etc/sendmail.cf**, то делается жутко от одного только размера в 32 Кбайта (более 1000 строк). А если заглянуть внутрь файла, то становится еще страшнее от непонятных ключей и директив.

В файле конфигурации sendmail все параметры сгруппированы по разделам. Разбиение оформляется в виде следующих строк:

```
##################
#  local info  #
##################
```

Такой комментарий указывает на то, что далее идет раздел `local info`. Таких секций несколько, например:

- 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, тем более что конфигурационный файл в моей системе занимал 50 Кбайт. Если расписывать каждый параметр, понадобится отдельное издание. Наша задача — эффективность и безопасность, поэтому рассмотрим только настройки, касающиеся этих вопросов, и испытаем sendmail в действии.

Для упрощения конфигурирования sendmail в последних версиях этого почтового сервиса используется новый файл — sendmail.mc, который можно найти в директории **/etc/mail**. Пример содержимого файла можно увидеть 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
file,
dnl you need the sendmail-cf rpm installed and then have to generate a
dnl new /etc/sendmail.cf by running the following command:
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 sent
out
dnl through an external mail server:
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, чем старый sendmail.cf, благодаря чему уменьшается вероятность допустить ошибку при конфигурировании. После исправлений sendmail.mc необходимо конвертировать в формат CF специальной командой, в результате чего получается файл **/etc/sendmail.cf**.

Мы в основном будем рассматривать параметры, которые нужно устанавливать в файл sendmail.cf, а если описывается параметр sendmail.mc-файла, то я буду явно указывать на это.

В *гл. 2* я уже упоминал, что 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 только одна строка:

```
127.0.0.1    localhost.localdomain    localhost
```

Подробно этот файл будет описан в *гл. 11*, когда мы будем 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.

Программа sendmail использует имя компьютера, которое вы указали во время установки ОС Linux. Выполните команду `hostname`, чтобы узнать его. В моем случае это FlenovM. Так уж получилось, что все сетевые обращения происходят не по имени, а по IP-адресу, а sendmail не может определить адрес по имени FlenovM, поэтому происходит зависание. Чтобы исправить эту ситуацию, добавьте в файл **/etc/hosts** строку:

```
192.168.77.1    FlenovM    FlenovM
```

Только FlenovM нужно заменить на имя вашего компьютера и указать свой IP-адрес. После этого программу sendmail можно помещать в автозапуск, и она будет работать великолепно даже с настройками по умолчанию.

Каждому пользователю системы автоматически создается почтовый ящик, и если сервис sendmail запущен, то им уже можно пользоваться. Все почтовые ящики хранятся в директории **/var/spool/mail/**. Их имена соответствуют именам пользователей. Так для учетной записи root ящик будет расположен в папке **/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 я насчитывал до 7 различных клиентов. Какой выберете вы, зависит только от личных предпочтений.

Можно обойтись и без клиента, а соединение с сервером осуществлять напрямую через сервис Telnet. Благо команды SMTP достаточно простые и их легко использовать. В этом случае для отправки почты нужно подключиться к 25 порту (SMTP-порт), а для получения — к 110 (порт POP3).

8.2. Работа почты

Рассмотрим работу ящиков на примере почтового клиента KMail — графическая программа, которая у вас должна быть, если установлен KDE. Для ее запуска из главного меню Linux выберите строку **Internet/KMail**. Перед вами откроется окно, как на рис. 8.1.

Программа еще не знает, с каким почтовым ящиком мы хотим работать, это необходимо настроить. Нажмите меню **Settings/Configure KMail**. Перед вами откроется окно конфигурации. Выберите раздел **Network**, и вы увидите окно из двух вкладок: **Sending** и **Receiving**.

На вкладке **Sending** мы должны указать параметры сервера, через который будет происходить отправка. По умолчанию уже настроен локальный sendmail, но если почтовый сервер расположен на другом компьютере? Давайте удалим существующую запись (выделите и нажмите кнопку **Remove**), а потом создадим свою.

Нажмите кнопку **Add** для добавления новой записи. Перед вами появится окно выбора протокола: SMTP или Sendmail. Выбираем SMTP, как наиболее универсальный вариант. Вслед за этим откроется окно, в котором нужно указать параметры SMTP-сервера, через который мы будем посылать почту (рис. 8.3).

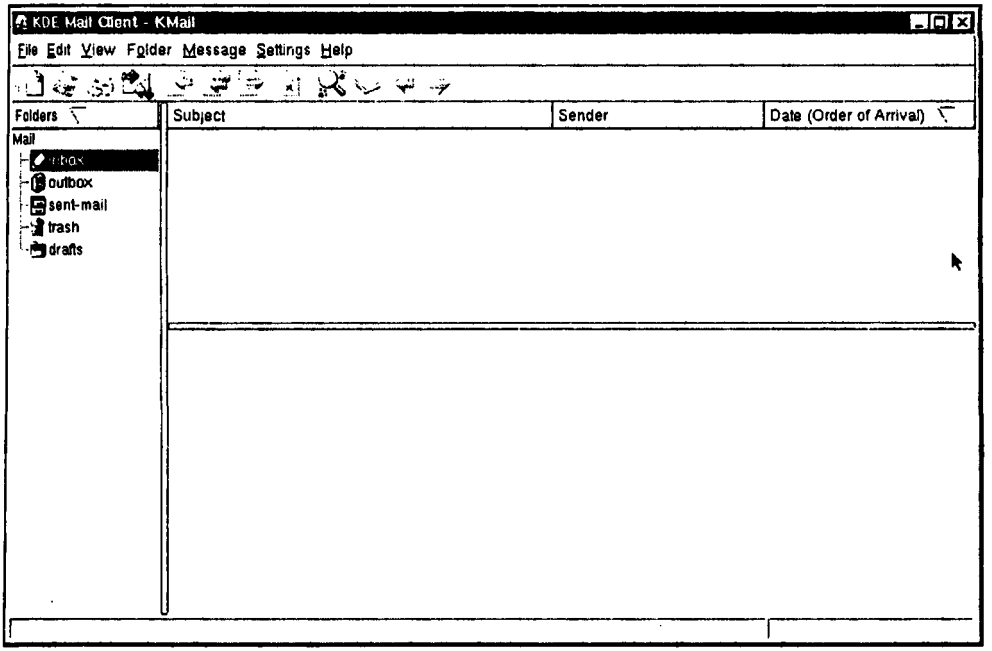

Рис. 8.1. Главное окно программы KMail

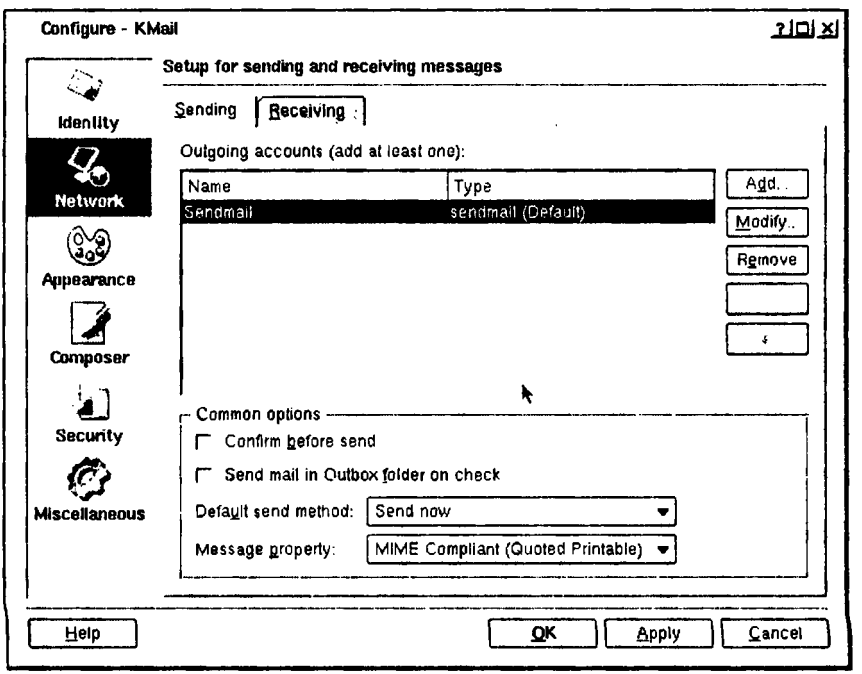

Рис. 8.2. Окно сетевых настроек

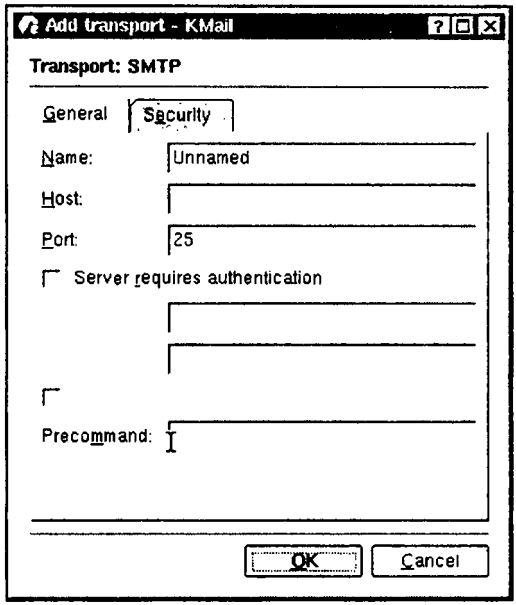

Рис. 8.3. Окно настроек SMTP-сервера

В этом окне нам нужно заполнить следующие поля:

- **Name** — имя сервера, которое может быть любым;
- **Host** — адрес SMTP-сервера. Если вы используете локальный сервер, то можно указать localhost или 127.0.0.1;
- **Port** — порт SMTP-сервера. Чаще всего используется 25 порт, но это значение может быть изменено;
- если сервер требует аутентификации, то поставьте галочку в **Server requires authentication** и заполните открывшиеся поля **Login** и **Password**.

Если вы не первый день в Интернете и работали с электронной почтой, то процесс создания параметров SMTP-сервера не должен вызвать проблем.

Теперь рассмотрим настройку сервера для чтения почты. Перейдите на вкладку **Receiving**, и вы увидите список серверов. Выделите имеющиеся записи и удалите. Нажмите кнопку **Add**, чтобы добавить сервер, через который мы будем получать почту. Перед вами откроется окно, в котором нужно выбрать тип сервера: **Local mailbox**, **POP3**, **IMAP**, **Maildir Mailbox**. Чаще используется POP3, и процесс его создания схож с SMTP-сервером. Вы также должны указать адрес сервера, порт (по умолчанию 110) и имя с паролем.

Наиболее интересным может оказаться использование локального ящика. Даже если SMTP-сервер не установлен, в системе создается директория с ло-

кальным почтовым ящиком, куда для администратора root помимо привычных E-mail попадают извещения по безопасности. Если вы работаете в консоли и увидели сообщение типа "You have new mail", то это означает, что в ваш ящик в локальной директории попало новое сообщение. Для его проверки лучше всего использовать почтовый клиент.

Итак, создадим новую учетную запись, чтобы можно было в удобном виде читать сообщения по безопасности. Щелкните кнопку **Add**, и перед вами появится окно выбора типа сервера. Укажите тип ящика **Local mailbox** и нажмите кнопку **OK**. Вы увидите окно создания нового аккаунта, как на рис. 8.4.

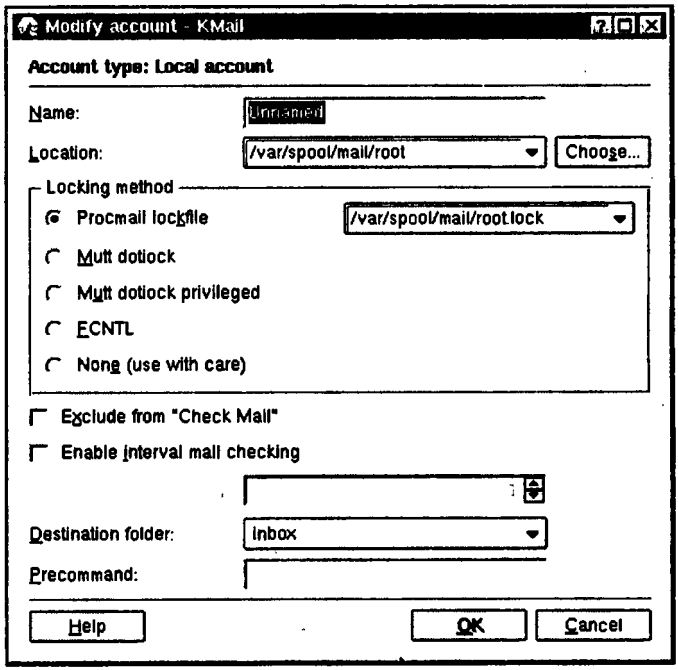

Рис. 8.4. Окно настроек аккаунта Local Mailbox

Здесь необходимо заполнить следующие поля:

- **Name** — имя записи, может быть любым;
- **Location** — расположение локального ящика. По умолчанию все ящики хранятся в директории **/var/spool/mail/имя**, где имя — это имя пользователя. Для администратора root размещение будет **/var/spool/mail/root**.

Остальные параметры, чаще всего, остаются заданными по умолчанию, если администратор не натворил чего-то особого в конфигурации.

Попробуйте прочитать почту разными протоколами. Убедитесь, что сообщения приходят на ваш почтовый ящик и доходят до получателя. Даже с настройками по умолчанию все должно работать верно. В дальнейшем мы рассмотрим некоторые специфичные настройки, которые помогут сделать ваш почтовый сервер более безопасным, но прежде чем приступать к улучшениям, нам нужно удостовериться, что работает базовый вариант.

8.2.1. Безопасность сообщений

E-mail-сообщения пересылаются по сети в виде чистого текста. Если злоумышленник перехватит такое сообщение, то без проблем сможет его прочитать. При передаче конфиденциальной информации необходимо использовать шифрование.

Наиболее распространенными методами шифрования на данный момент являются:

- S/MIME (Secure/Multipurpose Internet Mail Extensions, безопасные многоцелевые расширения электронной почты в сети Интернет) — этот стандарт шифрования поддерживается в основном почтовыми клиентами Netscape и его клонами. Это накладывает некоторые ограничения, ведь не все пользователи привыкли пользоваться именно этими программами;
- PGP — программа шифрования, используется во многих областях, в том числе и в почтовых сообщениях. Этот стандарт поддерживает большое количество почтовых клиентов. Существует несколько реализаций PGP, но многие специалисты рекомендуют использовать программу GnuPG. Нет, она не лучше других, потому что все PGP-клоны используют один и тот же принцип. Просто GnuPG разработана за пределами США, где уже не действует закон об ограничении длины ключа.

Таким образом, мы шифруем текст сообщения. Но сам протокол работает без шифрования, поэтому все пароли передаются по сети в чистом виде, и их тоже необходимо защитить. Для этого можно использовать один из современных стандартов RFC 1734 (MD5 APOP Challenge/Response), RFC 2095 (MD5 CRAM-HMAC Challenge/Response) или прибегнуть к помощи stunnel.

8.3. Полезные команды

Давайте рассмотрим некоторые команды, которые помогут вам в администрировании sendmail-сервера:

- `hoststat` — показать состояние хостов, которые недавно работали с локальным почтовым сервером. Команда является эквивалентом `sendmail -bh`, которая по умолчанию неактивна;

- `mailq` — отобразить краткую информацию о сообщениях в очереди, ожидающих обработки. Пример результата выполнения команды:

```
/var/spool/mqueue (1 request)
----Q-ID---- --Size-- -----Q-Time----- -----Sender/Recipient-----
j0IAnST11838  6   Tue Jan 18 13:49  <flenov@flenovm.ru>
        (host map: lookup (flenovm.ru): deferred)
        <root@flenovm.ru>
```

  Из первой строки видно, что в очереди находится одно сообщение. Вторая строка включает дату посылки и адрес отправителя — `flenov@flenovm.ru`. В последней строке отображается получатель сообщения — `root@flenovm.ru`;

- `mailstats` — отобразить статистику сообщений, включая количество переданных байт;

- `sendmail` — это команда самого сервера sendmail. Запуская ее с различными ключами, можно увидеть достаточно много полезной информации. За более подробной справкой следует обратиться к документации man.

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 насмехательств. Я слышал, что некоторые даже делали ставки на то, будет найдена ошибка в этом месяце или нет.

Как я и обещал, в данном разделе мы поговорим о некоторых параметрах, повышающих безопасность.

8.4.1. Баннер-болтун

Проблема безопасности начинается 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 об этих данных. Для этого 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 — это ключ `$v/$Z`, который отображает версию и само имя `Sendmail`. Именно поэтому он был убран в последних версиях.

Теперь значение этому параметру присваивается следующим образом:

```
SmtpGreetingMessage=$j $b
```

Если в вашей системе отображается что-то лишнее, то это 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 Web-серверах sendmail может стоять для того, чтобы прямо из сценариев на Perl или PHP можно было отсылать письмо. Если ваш сервер не будет принимать писем, то необходимо запретить этот режим. 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 на `-bd`.

Если у вас нет директории **/etc/sysconfig/sendmail** (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 директории и файлы. В sendmail это возможно сделать, и в последних версиях уже реализовано с помощью параметра `RunAsUser`:

```
O RunAsUser=sendmail
```

По умолчанию эта строка может быть закомментирована. Уберите комментарий. Можно также явно добавить описание группы, с правами которой должна происходить работа:

```
O RunAsUser=sendmail:mail
```

В данном случае `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 перечислить все параметры через запятую:

```
O PrivacyOptions=authwarnings,noexpn,novrfy
```

Наиболее опасной может оказаться для сервера опция VRFY, которая позволяет проверить существование почтового ящика. Именно ее запрещает третья строка в данном примере.

Вторая строка устанавливает параметр `noexpn`, запрещающий команду EXPN, которая позволяет по почтовому псевдониму определить E-mail-адрес или даже имя пользователя. С помощью этой директивы хакеры могут собирать себе 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 sendmal работать через безопасный интерпретатор команд. Для этого специально был разработан smrsh. Чтобы почтовый сервис использовал именно его, проще всего добавить следующую строку в файл sendmail.mc:

```
FEATURE('smrsh', '\bin\smrsh')
```

В данном случае в скобках указано два параметра: имя командного интерпретатора и каталог, в котором он располагается. Убедитесь, что в вашей системе именно такой путь, или измените параметр.

По умолчанию интерпретатор smrsh выполняет команды из каталога **/usr/adm/sm.bin**. Программы из других директорий запускать невозможно. Если в каталоге **/usr/admsm.bin** находятся только безопасные программы, то ваша система будет менее уязвима.

8.4.6. Доверенные пользователи

В сервисе sendmail можно создать список пользователей, которым вы доверяете отправлять 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 Web-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. Таким образом, подключение на 25 и 110 порты, 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 надобности не стоит, только если вы уверены в производительности сервера.

8.5. Почтовая бомбардировка

С почтовой бомбардировкой я встретился первый раз почти 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 ради шутки забросал меня почтовыми бомбами.

Что такое почтовая бомба? Это простое письмо с бесполезным содержимым любого размера. Хакеры забрасывают свою жертву такими посланиями, чтобы переполнить ящик и лишить его возможности принимать другие сообщения. Классическая задача DoS, только в отношении почтового ящика.

На первый взгляд, необходимо просто увеличить размер ящика или убрать лимит вовсе. Это самое неверное решение, поэтому забудьте про него. Если ящик не будет иметь предела, то хакер сможет произвести атаку DoS на сервер.

Почтовые сообщения — единственный способ закачать информацию на сервер. Когда злоумышленник отправляет E-mail-письмо, оно сохраняется там, 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 директории по умолчанию (раздел **/var**). Если этот раздел будет переполнен, то сервер не сможет больше записывать в него информацию. В разделе **/var** хранятся еще и журналы безопасности. Если они не смогут пополняться, то сервер окажется полностью недоступным.

Ограничения 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 задачу злоумышленника. Для этого нам понадобятся параметры, которые мы рассматривали в *разд. 8.4.7*. Помимо этого, желательно ограничить максимальный размер сообщения с помощью параметра `MaxMessageSize` до приемлемых пределов. Таким образом, злоумышленнику придется направлять на сервер множество маленьких писем вместо нескольких больших.

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 нашего 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 использование общедоступных или взломанных в Интернете серверов.

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. Благодаря этому хакер может даже провести атаку DoS на почтовый сервис.

8.6.1. Блокировка приема спама

Прием нежелательной корреспонденции приводит к следующим негативным последствиям:

- как мы уже говорили, излишние расходы на трафик, которые постоянно увеличиваются;
- отвлекает внимание сотрудников вашей организации 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, и я надеюсь, что описанных ваше уже достаточно, чтобы вы начали предпринимать какие-либо 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(dnssbl, 'spam.domean.com', ' 550 Mail not accepted from this
domain')dnl
```

Версия 8.11:

```
HACK('check_dnsbl', 'spam.domean.com', '', 'general', 'reason')dnl
```

В обеих строках `spam.domean.com` нужно заменить на адрес домена, с которого рассылается спам, чтобы заблокировать соединение с ним. Этот метод не очень эффективен, потому что 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. Таким образом, некоторые пользователи не могли работать с нашими программами. Такое продолжалось в течение месяца, пока не убедились, что спам-листы перестали быть эффективными, и стали искать другие методы.

Фильтрация сообщений

Более точный метод — блокировка сообщений по их содержимому. Специализированная программа анализирует всю информацию, которая проходит через сервер, и ищет характерные признаки спам-рассылки. Если письмо определяется как спам, то оно удаляется.

Этот способ более эффективный, но по тексту очень сложно определить, является ли письмо рекламной рассылкой. Хакеры постоянно ищут новые пути обхода таких блокировок, поэтому процент фильтрации достаточно невысок. Вы можете настроить программу так, что она будет уничтожать все сообщения, в которых есть слова "купи", "продаю" и тому подобные термины, характерные для спама, но тогда могут быть удалены и необходимые вам письма.

Я не буду советовать никаких программ фильтрации нежелательной почты, потому что не вижу идеального решения. Но если вы захотите использовать подобный способ, то хочу только обратить ваше внимание на программу Spamassassin (**http://spamassassin.apache.org/**). В ней реализовано множество проверок, которые позволяют достаточно эффективно определять нежелательные сообщения.

Помимо этого, может пригодиться изменение параметра `MaxRcptsPerMessage` (команда sendmail сервера), который устанавливает максимальное количество получателей сообщения. Если их более 100, то это однозначно указывает на спам. Хотя в некоторых организациях используются почтовые рассылки всем сотрудникам, которые могут содержать до 1000 адресатов. В этом случае может быть уничтожено очень важное письмо, поэтому необходимо, чтобы администраторы отправляли письма не более чем 20 получателям за один раз.

8.6.2. Блокировка пересылки спама

При конфигурации почтового сервиса вы должны сделать так, чтобы злоумышленники не смогли посылать свой спам через ваш сервер. Вам необходимо произвести несколько настроек, чтобы массовая рассылка перестала быть эффективной:

- по умолчанию протокол SMTP не требует авторизации, поэтому любой пользователь может подключиться к серверу и отправить письмо. Чтобы избежать этого, можно выполнить одно из следующих действий:
  - запретить с помощью сетевого экрана подключение к SMTP-порту пользователей, которые находятся вне вашей сети. Эту защиту чаще всего используют провайдеры и администраторы частных или корпоративных сетей. С реализацией этого метода 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 приема почты по протоколу POP3. В момент проверки почты сервер производит авторизацию клиента, и по этим данным может временно создаваться разрешающая запись в сетевом экране (или другим способом) для доступа к SMTP. Теперь в течение 10 минут с этого IP-адреса можно проверять почту;
  - использовать авторизацию SMTP. Изначально в стандарте на протокол отправки сообщений нет ничего об идентификации пользователя, поэтому не все серверы поддерживают ее. Но в sendmail и других мощных пакетах есть расширение, которое позволяет сделать авторизацию;
- 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, список которых может содержаться в поле "CC".

Существуют и другие методы, но даже этих будет достаточно.

В последних версиях sendmail уже 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 для рассылки почты только с компьютеров внутренней сети или 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 уже 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 очень 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 (**http://popsmtp.sourceforge.net**), который проверяет журнал сообщений **/var/log/maillog**, и если в нем найдена удачная авторизация за определенный период, то отправка почты разрешена.

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 proxy/Firewall, также автоматически считаются авторизованными. Таким образом, поиск по IP-адресу записей в журнале не может дать 100 % 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. Заключение

Работу электронной почты мы рассмотрели поверхностно, потому что конфигурирование sendmail требует отдельной книги. В документации, которая идет с ОС (ее можно найти в каталоге **/usr/share/sendmail-cf**), вы можете

прочитать достаточно подробные инструкции по настройке сервиса. Если в любой поисковой системе запустить поиск по контексту "Конфигурирование Sendmail", то перед вами откроется большой список материалов по этой теме.

Если этого окажется недостаточно, то можно приобрести книгу по данной теме, например, Sendmail, авторы Bryan Costales, Eric Allman, издательство O'Reilly.

Если вы только начинаете изучать почтовый сервис sendmail, то я рекомендую отталкиваться от конфигурационного файла по умолчанию.

ГЛАВА 9

Шлюз в Интернет

Мы уже знаем, как установить и настроить Linux-сервер, сделать его соединение безопасным и подключить основные сервисы. Но такие службы, как Web-сервер и электронная почта, используются не только в локальной сети. Их максимальные преимущества проявляются при интеграции со всемирной сетью.

Но Интернет небезопасен. Там можно встретить разных людей и столкнуться с реалиями жизни. Точно так же в Интернете могут быть законопослушные пользователи, а могут быть и взломщики.

Когда мы строим дом, то уже при возведении стен заботимся о безопасности, устанавливая двери, думаем об их надежности, а когда заканчиваем 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 входить в сеть с улицы (из Интернета). После завершения всех настроек мы повесим на нашу обитель сигнализацию — системы мониторинга и выявления атак, которые будут рассматриваться в *гл. 12*.

В качестве дверей во всемирную сеть будут выступать шлюз и прокси-сервер. 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 предстоит познакомиться.

9.1. Настройка шлюза

Выход в Интернет можно осуществить через модем или выделенную линию. В *разд. 3.7* мы уже бегло рассмотрели возможность подключения с помощью

графической утилиты, позволяющей настроить соединение обоих типов. Мы не будем останавливаться на этом вопросе, потому что здесь мне нечего добавить по поводу безопасности. Настройка хорошо описана во множестве документов, которые легко найти в Интернете. Но несколько замечаний я бы хотел сделать.

Используя графическую утилиту, вы должны знать, что все сценарии находятся в директории **/etc/ppp** и **/etc/sysconfig/network-scripts**. Обязательно ознакомьтесь с файлами, которые здесь находятся.

Подключившись к Интернету, убедитесь, что вам доступен DNS-сервер, который находит соответствие символьных имен и IP-адресов. Для этого можно выполнить команду ping с указанием какого-либо сервера, например:

```
ping www.redhat.com
```

Если ответ получен, то сервис преобразования имен доступен, иначе с Интернетом можно будет работать, используя только IP-адреса. Если имя не определяется, то ситуацию можно изменить, вручную прописав адрес DNS-сервера. Получите эту информацию у своего интернет-провайдера и добавьте следующую строку в файл **/etc/resolv.conf**:

```
nameserver 191.168.1.1
```

Адрес 191.168.1.1 нужно заменить на тот, что вам предоставил провайдер. Если вам дали несколько адресов, то можно указать их все, каждый в своей строке:

```
nameserver 191.168.1.1
nameserver 191.168.1.2
```

9.2. Работа прокси-сервера

Изначально прокси-серверы (proxy) создавались для решения узкого круга задач, а именно, — кэширование данных, получаемых из Интернета. Например, вы работаете в сети из 100 компьютеров, которые подключаются к Интернету через один физический канал связи. Не секрет, что большая часть пользователей загружает по несколько раз в день одни и те же страницы, и каждый раз эта закачка давит на канал связи и трафик.

Сделаем простейший расчет. Ежедневно мы обращаемся к поисковой системе, например, **www.yahoo.com** или **www.google.com**. Теперь посчитайте, сколько происходит загрузок сайта **www.yahoo.com** со 100 компьютеров. В результате должно получиться число более 1000, потому что в среднем человек просматривает не менее 10 страниц за счет поиска разной информации и уточнения запросов. Таким образом, трафик расходуется напрасно.

Для экономии интернет-трафика были придуманы прокси-серверы. Со временем их возможности стали наращиваться, и на данный момент 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 данной главе мы поговорим о самом популярном в Linux прокси-сервере squid, рассмотрим его возможности, оценим безопасность и поговорим о конфигурировании.

Чтобы сэкономить трафик и одновременно увеличить скорость загрузки, на сервере, через который осуществляется выход в Интернет, устанавливается специализированная программа proxy, которая кэширует весь трафик (рис. 9.1). Когда первый пользователь загружает страницу **www.yahoo.com**, то все ее содержимое сохраняется в кэше proxy. При следующем обращении к Web-узлу все картинки скачиваются уже не из Интернета, а с прокси-сервера провайдера, а текстовая часть (в зависимости от содержимого страницы и происшедших изменений) может быть загружена с сервера.

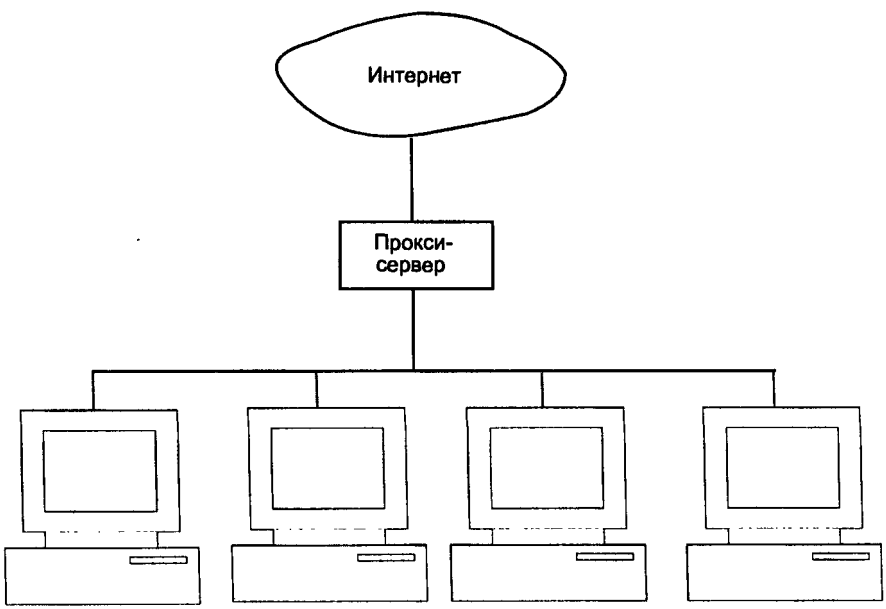

Рис. 9.1. Организация работы с Интернетом через прокси-сервер

Как правило, на сайтах именно графический материал занимает наибольший объем. Если текстовая часть страниц обычно не превышает 15 Кбайт, то размер графики достигает 100 Кбайт. Загружая эту информацию с локального прокси-сервера, вы экономите трафик и время.

Скорость загрузки увеличивается за счет того, что proxy находится в вашей локальной сети, и связь с ним, чаще всего, 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 и неизмененную текстовую часть). Связь с Интернетом намного ниже, и в малых офисах в среднем составляет от 2 до 8 Мбит/с. Через этот канал вы забираете только измененные текстовые данные (чаще всего, содержимое HTML-файлов).

Помимо кэширования содержимого страниц, proxy может сохранять результаты DNS-запросов. Это также может повысить производительность. Пользователю удобнее вводить символьные адреса, а компьютер обменивается данными через IP-адрес. Исходя из этого, прежде чем начнется загрузка, программа должна выполнить такую подмену. Это занимает какое-то время и создает задержку перед началом обмена данными. Если до вас уже кто-нибудь обращался к сайту по символьному имени, то задержки на работу с сервером DNS не будет, потому что proxy возьмет адрес из своего кэша. Более подробно о DNS мы поговорим в *гл. 11*.

С развитием всемирной сети и увеличением потребностей пользователей стали наращиваться и возможности proxy. Теперь прокси-сервер может выполнять роль шлюза и обеспечить доступ в Интернет без дополнительных программ или оборудования. Помимо этого, он становится щитом в сети от вторжения извне. Например, все пользователи подключены к Интернету через proxy. При этом сервер прячет реальный IP-адрес пакетов и отправляет их в сеть от своего имени, т. е. хакеры видят только IP-адрес прокси-сервера и будут ломать его, а компьютеры реальных пользователей останутся незатронутыми. Таким образом, намного проще при организации защиты от внешнего вторжения больше внимания уделять охране прокси-сервера, чем клиентским компьютерам. Но несмотря на это, возможности proxy с точки зрения защиты слишком простые и их легко можно обойти, поэтому без хорошего сетевого экрана и зоркого глаза администратора все же не обойтись.

Возможность сокрытия IP-адреса дает еще одно преимущество — экономия адресов. Интернет-адрес должен быть только у прокси-сервера, потому что 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 прозрачные и анонимные. Прозрачные пакеты пользователя (без изменения адреса отправителя) просто пересылаются дальше на

Web-сервер. Proxy, 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.

На данный момент в Интернете работает множество анонимных прокси-серверов, но только 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.

Так как не все компьютеры в сети должны иметь право работать с Интернетом, то на уровне прокси-сервера можно производить аутентификацию пользователя.

В некоторых версиях proxy есть очень удобная возможность — обмен информацией между серверами. Например, в здании в одной большой сети находятся несколько офисов, но каждый из них платит за Интернет отдельно, поэтому общается с внешним миром через свой прокси-сервер. Можно объединить proxy, и если на одном нет в кэше нужного сайта, то он возьмет информацию с соседнего сервера.

Чаще всего для реализации такой возможности применяется ICP-протокол (Internet Cache Protocol, протокол интернет-кэширования). Если ваш сервер не нашел нужного документа, то он направляет ICP-запрос другим серверам. Если какой-либо proxy ответит положительно, то информация будет взята у него.

При использовании протокола ICP (или иного способа поиска данных в других proxy) выигрыш от скорости загрузки становится не столь значительным при обращении к документам маленького размера, потому что увеличивается время на ICP и поиск информации в кэше. При большой нагрузке на серверы и немалой базе кэша поиск может оказаться слишком долгим, и скоростное преимущество исчезает. Единственное, что остается в вашем распоряжении — экономия трафика, которая может сберечь деньги тем, кто оплачивает каждый получаемый мегабайт.

Мы рассмотрели основные возможности proxy, но это не значит, что все они есть в любом сервере. Все зависит от разработчика, а некоторые реализуют только одну задачу.

Для работы через прокси-сервер вы должны настроить соответствующую программу, например, браузер Mozilla. Запустите этот обозреватель и выберите меню **Edit/Preferences**. В появившемся окне с левой стороны расположен список категорий для конфигурирования. Выберите **Advanced/Proxies**, и

перейдите к настройке подключения через прокси-сервер. По умолчанию установлено автоматическое определение соединения (**Direct connection to the Internet**). Вы должны поменять этот параметр на ручную конфигурацию (**Manual proxy configuration**) и указать IP-адрес и порт прокси-сервера для каждого протокола (рис. 9.2).

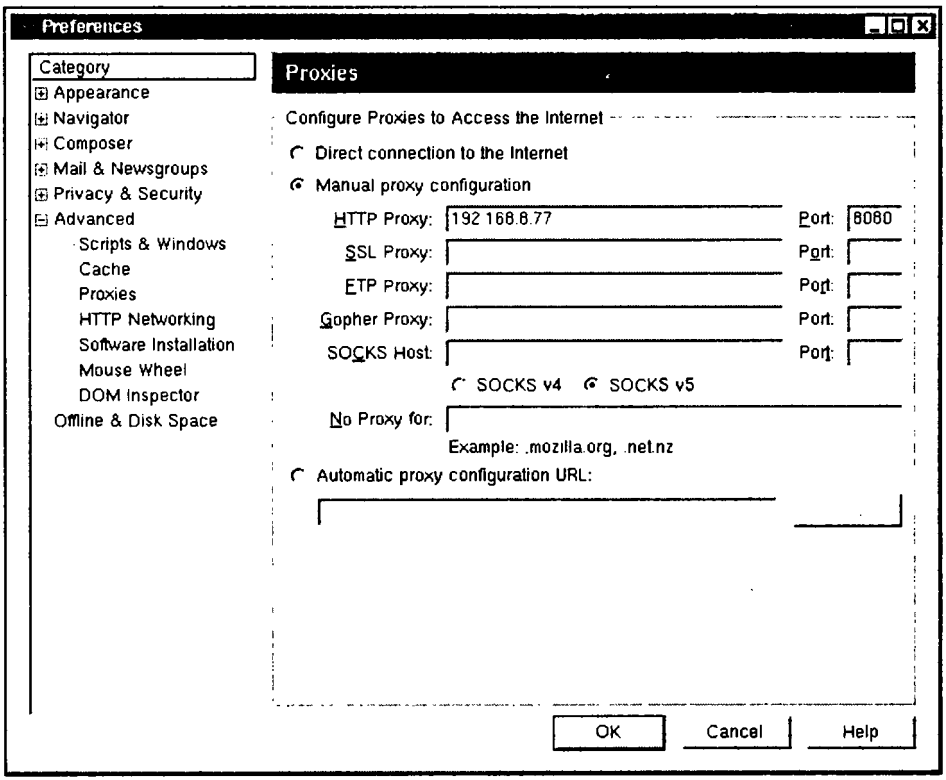

Рис. 9.2. Настройка соединения через прокси-сервер в браузере Mozilla

После этой настройки браузер будет посылать все запросы прокси-серверу, а тот уже перенаправит их серверу. Proxy постоянно должен находиться в загруженном состоянии и прослушивать определенный порт (или несколько портов для разных протоколов).

Под каждую задачу, поддерживающую определенный протокол, как правило, выделяется отдельный порт. Для HTTP-протокола, применяемого для загрузки Web-страниц, чаще всего используется порт 8080, но это значение зависит от сервера и может быть изменено. Перед использованием определенной программы прокси-сервера убедитесь, что она обладает необходимыми вам возможностями и обеспечивает поддержку всех нужных протоколов. Неподдер-

живаемые протоколы придется направлять, минуя сервер, т. е. напрямую через шлюз.

Для повышения безопасности вашей сети необходимо запретить с помощью сетевого экрана подключения извне на используемые сервисом squid порты. Например, для его работы с HTTP-протоколом по умолчанию используется порт 3128. И если к этому порту будет разрешено подключаться только из локальной сети, то хакер не сможет применять этот прокси-сервер в своих целях или для получения доступа к компьютерам этой сети.

9.3. squid

Как я уже сказал, самым распространенным прокси-сервером является squid. Этот сервер имеет достаточно длинную историю, и за время его существования в нем реализовано много возможностей. Еще не было ничего такого, что я не смог бы получить с помощью squid.

Основной конфигурационный файл для squid — **/etc/squid/squid.conf** (в некоторых системах место его расположения **/etc/squid.conf**). Файл очень большой, и приводить его полностью нет смысла, т. к. значительную его часть занимают подробные комментарии по использованию директив.

Рассмотрим основные команды, которые нам доступны для управления прокси-сервером. Как обычно, все параметры, влияющие на производительность и безопасность, мы разберем подробно. Остальные настройки будут рассмотрены поверхностно, с ними можно поближе познакомиться по комментариям из конфигурационного файла.

9.3.1. HTTP-директивы

При подключении к Интернету пользователи первым делом стремятся загрузить Web-страничку. Если используется proxy, то необходимо правильно настроить HTTP-протокол. Для решения этой задачи в squid есть следующие директивы:

- `http_port n` — параметр `n` определяет номер порта, через который будет происходить подключение.

  Первое, что нам необходимо настроить, — это порты, на которых сервер будет ожидать подключения клиентов. Такие директивы имеют формат `xxxx_port`. Для порта HTTP запись будет выглядеть таким образом:

  ```
  http_port 8080
  ```

  После этого при конфигурировании браузера на клиентском компьютере вы должны будете указывать IP-адрес сервера, где установлен squid, и выделенный в данной директиве порт;

- `hierarchy_stoplist` — определяет перечень URL-адресов, данные по которым 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 Web-страницу **www.servername.com/cgi-bin/ping.cgi**, на которой можно через Web-интерфейс выполнить директиву `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 кэшировать такие страницы, то вы всегда будете видеть одно и то же. В результате вы получите только неудобства от соединения через proxy.

  Вопросительный знак очень часто используется для передачи параметров в сценарии PHP, поэтому такие страницы тоже не рекомендуется 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
```

  Я думаю, вы согласитесь со мной, что незачем кэшировать то, что будет получаться с сервера, и зря расходовать дисковое пространство.

9.3.2. FTP-директивы

Для работы по FTP-протоколу тоже есть несколько директив:

- `ftp_passive параметр` — режим работы. Если в качестве параметра указано значение `on`, то разрешен пассивный режим (устанавливается по умолчанию).

  Сервер squid позволяет работать с FTP-протоколом, но может потребоваться некоторая 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 E-mail-адрес, 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@
  ```

  Правда, по умолчанию в файле **/etc/squid/squid.conf** эта строка закомментирована, но желательно поменять в ней E-mail-адрес, 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, будет зависеть удобство работы через прокси-сервер, поэтому я постараюсь показать все директивы, которые относятся к этому разделу, и подробно рассмотреть каждую из них:

- `cache_dir тип директория размер L1 L2 опции` — определяет параметры директории, в которой будет храниться кэш. Основными для нас являются `тип`, `директория` и `размер`. В большинстве случаев для типа применяется значение `ufs`, но если вы используете асинхронный ввод/вывод (я не советую, потому что вызывает проблемы в работе), то может быть установлено `aufs`.

  В качестве директории вы должны выбрать такую, которая находится в самом большом разделе, чтобы информация не разобщалась по нескольким дискам. Если у вас используется один диск с одним разделом, то расположение не имеет особого значения.

  Размер директории по умолчанию равен 100 Мбайтам. 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 Гбайта кэша. Выделенное пространство быстро исчезает, если серверу разрешено кэшировать большие файлы.

- `cache_mem n MB` — задает максимальный размер оперативной памяти, необходимый для программы. По умолчанию используется n=8 Мбайт. Если ваш сервер решает задачи только proxy, то можно указать значение, равное разнице объемов оперативной памяти и памяти, необходимой для ОС. Например, если у вас ОЗУ 512 Мбайт, то для ОС в текстовом режиме 64 Мбайта будет более чем достаточно. Остальную память (448 Мбайт) можно отдать прокси-серверу — чем больше у него оперативной памяти, тем быстрее он сможет отвечать на часто запрашиваемые страницы;
- `cache_swap_low n` — процент заполнения кэша. Когда размер кэша превышает значение `n`, сервер начинает чистить его, 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, попадающего в кэш. По умолчанию установлено значение 0, при котором порог отсутствует;
- `maximum_object_size n KB` — максимальный размер объекта, который должен кэшироваться. По умолчанию стоит значение 4096 Кбайт, что соответствует 4 Мбайтам. Для повышения производительности сервера необходимо понизить это значение, но тогда вы можете потерять на расходовании трафика. Если экономия трафика стоит более остро, то значение `n` необходимо 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 используется 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 week
```

- `quick_abort_min n KB` — минимальный размер объекта, устанавливающий при обрыве соединения необходимость закончить его скачивание и полностью сохранить. Это позволяет сократить трафик и значительно 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`. Поставьте `-1`, чтобы отключить эту возможность;
- `quick_abort_max n KB` — максимальный остаток объекта, при котором закачка будет прервана в случае обрыва соединения. По умолчанию установлено значение 16;
- `quick_abort_pct n` — параметр аналогичен `quick_abort_min n`, но в данном случае указывается максимальный 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 этому же адресу по истечении этого времени, то копия из кэша не будет использоваться, а произойдет попытка вновь 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 используется 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 статичный адрес, который изменяется 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 не найден DNS-адрес, то это может быть проблема с самим сервером имен, а не сайтом. 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 нулевым, чтобы пользователи увидели нужный сайт сразу после устранения проблемы;

- `range_offset_limit n KB` — параметры кэширования. Если указать значение `-1`, то сервер загрузит из Интернета требуемый объект полностью, а потом будет транслировать пользователю полученные данные уже из собственного кэша. При значении `0` информация в запрошенном объеме будет передаваться между сервером и клиентом без кэширования на прокси-сервере, т. е. ни грамма лишнего. Если указано число более `0`, то proxy может кэшировать с интернет-сервера указанное количество килобайт. Например, пользователь запрашивает файл размером в 1 Мбайт и squid разрешено подкачивать 100 Кбайт, которые он сразу же может кэшировать. Это удобно, если пользователь не прервет передачу и не откажется забирать эту порцию, иначе получится, что сервер потратил трафик зря.

9.3.4. Журналы

В конфигурационном файле есть несколько параметров, влияющих на работу прокси-сервера с журналом (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 используется **/var/log/squid/cache.log**.;
- `cache_store_log файл` — журнал операций над объектами в кэше (убраны или помещены, на какое 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 нет утилит для анализа сохраняемых данных, да и пользы в них мало, только расходы на сохранение;
- `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 Linux и различных сервисах, 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 установлено 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 принято 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_port` в файле конфигурации squid. Атрибут `icp_порт` определяет порт, на котором настроен ICP-протокол, и соответствует параметру `icp_port` 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 тратить место в книге. 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. В случае если ответ не будет получен и придется обращаться в Интернет, пользователь ощутит большую задержку;
- `cache_peer_domain хост домен` — разрешить для хоста работу только с указанными доменами. Например, следующая строка позволит брать из кэша только то, что относится к домену **com**:

  ```
  cache_peer_domain parent.net .com
  ```

  Все остальные запросы будут игнорироваться, чтобы не перегружать сервер лишней работой. С помощью этого параметра можно настроить в сети несколько прокси-серверов, где каждый будет отвечать за свой домен.

9.3.6. Дополнительно

Рассмотрим оставшиеся параметры, которые я не смог отнести к определенным категориям, но представляющие для нас ценность:

- `redirect_rewrites_host_header параметр` — позволяет (`on`) или запрещает (`off`) изменять поле 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 IP-адрес для работы с Интернетом;
- `redirector_access список` — перечень запросов, проходящих через redirector. По умолчанию установлены все запросы;
- `cache_mgr email` — E-mail-адрес, на который будет послано письмо в случае возникновения проблем с работой proxy;
- `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 потом введет адрес redhat, то squid сам добавит имя домена, и вы попадете 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 proxy;
- `offline_mode параметр` — режим работы. Если параметр равен `on`, то squid будет взаимодействовать только с кэшем, и обращений к Интернету не будет. Если запрашиваемой страницы в кэше нет, то пользователь увидит ошибку. Чтобы squid мог адресоваться к Интернету, необходимо установить параметр `off` (установлено по умолчанию).

9.4. Права доступа к squid

Это самая больная 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 команду:

```
acl имя тип параметр
```

У данной директивы три параметра:

- имя — задается любым, и лучше всего, если оно будет описывать выполняемые действия;
- параметр — это шаблон или строка, смысл которой зависит от типа записи (второй аргумент);
- тип — можно указывать 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 авторизованные записи (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, M — Monday, T — Tuesday, W — Wednesday, H — Thursday, F — Friday, A — Saturday.

В файле конфигурации уже описано несколько правил, которые готовы к использованию и в большинстве случаев не требуют вмешательства. Их вы можете увидеть 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
```

Это список прав доступа, необходимых для минимальной работы прокси-сервера.

Рассмотрим первую строку. Здесь задается `ACl` с именем `all`. Так как используется тип шаблона `src`, то этому списку принадлежат юзеры, у которых IP-адрес соответствует 0.0.0.0/0.0.0.0, т. е. все пользователи.

Следующая строка создает ACL-запись 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 ACL-запись. Допустим, что 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. Всем остальным нужно запретить.

Уже при создании списка вы должны отталкиваться от идеи, что изначально доступ запрещен всем, и позволять 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 описания 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 — 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 данный 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 разрешили доступ к Интернету с IP-адресов только из диапазона 192.168.8.1—192.168.8.10, и если к прокси-серверу обратится компьютер с другим адресом, то в доступе 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 выход во всемирную сеть, и сделает нечто недозволенное.

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.

Внимание!

Аутентификация не работает, если squid настроен на работу в прозрачном режиме.

Однажды у некоторых работников оказалось большое превышение трафика. Вроде бы ничего удивительного, но настораживало то, что эти товарищи были в отпуске. Кто-то подделывал чужой адрес 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 устанавливается) и файл паролей. Если вы хотите использовать традиционную программу аутентификации proxy, то можно указать следующую строку:

  ```
  authenticate_program /usr/lib/squid/ncsa_auth /usr/etc/passwd
  ```

  Путь к программе ncsa_auth в вашей системе может отличаться.

  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 сможет производить проверку нескольких клиентов, поэтому если какой-либо 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 клиенты считают, что можно делиться паролем с друзьями. Не стоит этого делать, потому что за это отвечает 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 подключении Dial-up 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.

9.5. Замечания по работе 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 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 директорию с кэшем.

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 форсируемого сервера, который надо кэшировать, его порт и атрибуты сервера, который надо ускорять. Это задается с помощью следующих директив:

- `httpd_accel_host адрес` — адрес реального сервера;
- `httpd_accel_port порт` — порт Web-сервера. Чаще всего это порт по умолчанию (он равен 80), если не указан другой;
- `httpd_accel_uses_host_header параметр` — HTTP-заголовок включает в себя поле Host, не проверяемое сервером squid, 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 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 proxy.

Опять случай из собственной практики. Я снова вспоминаю фирму, где мне пришлось работать 4 года, и 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 произошло? 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 proxy. Так как все запросы шли от меня, то главный proxy ничего не заметил.

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 Netscrape/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
```

В данном случае `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 компью-

терам 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 будет не очень удобным. Некоторые сайты без графики теряют информативность, и с ними 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 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 информа-

цию. 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.

В первой строке описан шаблон, по которому запрещается загрузка адресов типа:

```
http://members.tripod.com/adm/popup/popup.html
```

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 достаточно большая и требует отдельной книги.

При борьбе с баннерами будьте готовы, что "обрезание" не всегда помогает, всплывающие окна могут снова появиться через определенное время. Это связано с тем, что баннеры — просто реклама, позволяющая зарабатывать деньги на существование сайта. Особо одаренные администраторы ищут любые возможные пути для того, чтобы ваша система не смогла избавиться от рекламы. Для этого постоянно изменяются адреса, чтобы регулярное выражение не сработало.

9.5.6. Подмена баннера

Пока что мы запретили загрузку баннеров или всплывающих окон. Но после этого Web-страницы перестанут 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 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 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 Web-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";
$various     = "$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 = $various 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;
```

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 эта программа заработала, создайте на своем Web-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 — HTML-файл, который будет закрывать всплывающие окна. В нем нужно поместить всего лишь функцию, которая будет закрывать окно. Для этого используется JavaScript-функция `window.close()`. Пример содержимого файла показан в листинге 9.3.

Все эти файлы должны лежать на Web-сервере в одной директории. Не забудьте в сценарии (в переменной `$YOURSITE`) указать правильный путь к этому каталогу.

Я постарался снабдить код в листинге 9.2 комментариями. Если у вас есть опыт программирования на Perl, то дальнейшие действия вы выполните без проблем.

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 уверен, но мне кажется, что он прав в том, что трафик с сайтов с интим-содержимым самый высокий (если не считать службу обновления Microsoft, где 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 трафика, но и другие непроизводительные расходы). Родители тоже не хотят, чтобы их дети рассматривали подобные сайты, поэтому стремятся оградить их от этого зрелища. Это я говорю уже как отец двоих детей.

Порно-сайты 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". А ведь это реальный случай из жизни, когда пользователь сети не смог попасть на сайт выставки по газовому оборудованию.

Создавать списки запрещенных сайтов достаточно сложно, но можно. В настоящее время в зоне **com** большинство сайтов эротической направленности закрылись и обживают другие места, которые принадлежат маленьким государствам. Существуют домены, которые на 90 % состоят из сайтов индустрии развлечений для взрослых. Вот их можно исключить полностью.

9.5.8. Ограничение канала

При организации доступа в Интернет очень часто требуется отдельным пользователям обеспечить большую скорость подключения. Как это сделать, когда по умолчанию все равноправны и могут работать на максимально доступной на данный момент скорости? Для этого нужно определиться с приоритетами. Для некоторых пользователей должен быть зарезервирован высокоскоростной канал связи. 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 уже заданы по умолчанию, и их следует заменить.

Давайте подробно рассмотрим 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 две строки, поэтому в первой параметр `n` равен `1`, а во второй — `2`.

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 и помимо этого для каждого пользователя индивидуально. В этом случае общий канал может быть понижен до 64 Кбит/с, и каждый пользователь в отдельности сможет работать только на скорости 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 работает 4 подсети, то каждой из них можно выделить по 64 Кбит/с, чтобы равномерно разделить нагрузку.

В нашем примере мы используем пулы 1 и 2 и снова 1 класса. 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
```

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
```

Так как пул 1 имеет 1 класс (`delay_class 1 1`), у которого можно ограничивать только канал полностью, в директиве используется единственный параметр — `скорость_канала` (`256000/256000`). Он формируется в виде двух чисел, разделенных знаком "/". До слэша указывается скорость, с которой будут скачиваться данные из сети, а после — размер пула, т. е. емкость, которую можно наполнить полученными из сети данными.

Количество параметров зависит от класса используемого пула. Если вы используете 1 класс, где можно ограничивать только общий канал, то должны быть указаны только два параметра:

```
delay_parameters пул скорость_канала
```

Если используется второй класс, то директива выглядит следующим образом:

```
delay_parameters пул скорость_канала индивидуально
```

Итак, первая директива использует полную скорость канала 256 000 байт в секунду. 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 списка.

В нашем примере для первого пула используется две строки:

```
delay_access 1 deny all
delay_access 1 allow admins
```

Сначала мы запрещаем доступ для всех, а потом разрешаем работать на данной 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 классу, то здесь нужно указать общую скорость (256 000 байт в секунду) и скорость доступа каждого компьютера в отдельности — 4000 байт 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 работать на скорости 4000 байт 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 мультимедийных файлов в рабочее время. В следующем примере мы разрешаем быстро читать Web-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, которое 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. Кэширование браузером

В заключение темы о прокси-серверах хочется сказать, что кэширование удобно, когда с сервером работает множество компьютеров. При этом экономнее используется канал связи, и с увеличением количества пользователей нагрузка на него увеличивается не так сильно, как могло бы быть.

Помимо централизованного кэширования с помощью прокси-сервера может использоваться локальный кэш сетевых программ. Например, в браузере Mozilla прочитанные страницы могут сохраняться в кэше на локальном диске. При следующем их вызове не происходит обращения даже к прокси-серверу, а страничка загружается из локального кэша.

На рис. 9.3 показано окно настройки кэша в браузере Mozilla. Как видите, **Memory Cache** по умолчанию составляет 4096 Кбайт — это максимальный размер оперативной памяти для хранения последних загруженных страниц. Использование кэша памяти увеличивает скорость, когда вы путешествуете по одному и тому же сайту, а при переходе от страницы к странице большинство графических образов сохраняется. Чтобы не делать даже лишних обращений к диску, последние загруженные файлы берутся прямо из оперативной памяти.

Еще один параметр отвечает за размер кэша на жестком диске (**Disk Cache**). Это максимальный размер директории, которая хранит загруженные из Интернета файлы. В моей системе значение по умолчанию равно 50 000 Кбайт (около 50 Мбайт). Это очень мало, и при регулярной работе с Web вы не заметите, как израсходуется это пространство. Если позволяет свободное место на жестком диске, то я рекомендую увеличить это значение. Еще ниже есть поле **Disk Cache Folder**, в котором указывается расположение папки для хранения кэша.

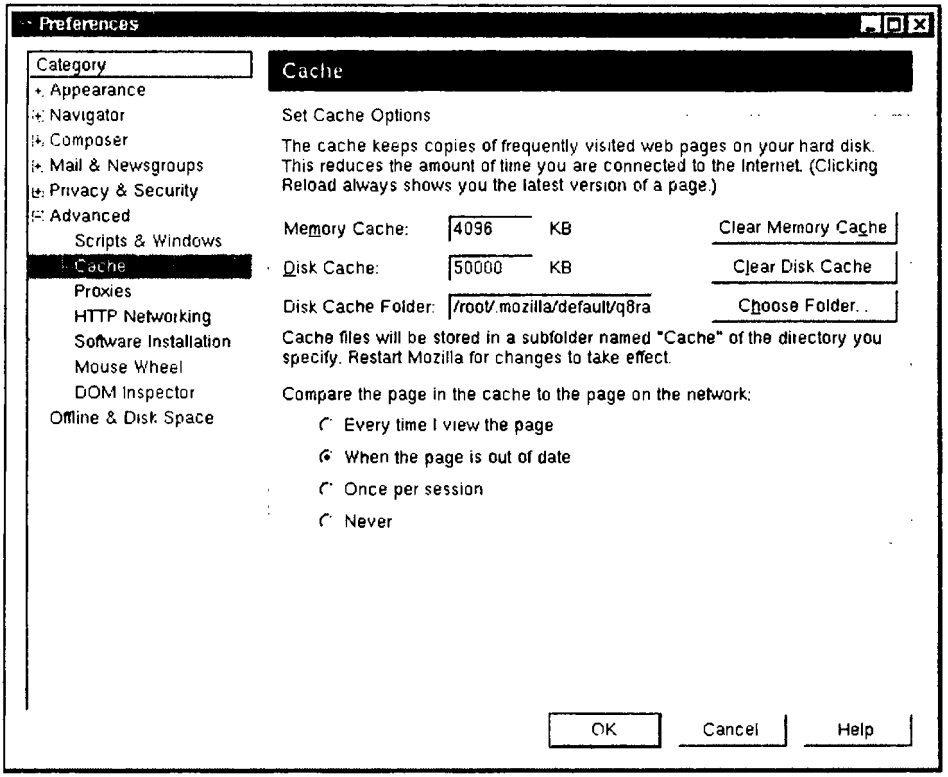

Рис. 9.3. Настройка кэширования в браузере Mozilla

И последнее, что можно настроить — условие, когда информация должна быть загружена с сервера, а не из кэша. Здесь можно выбрать одно из значений:

- **Every time I view the page** — обновление будет происходить каждый раз, когда вы обращаетесь к странице, а значит локальный кэш использоваться не будет;
- **When the page is out of date** — когда страница устарела;
- **Once per session** — единожды за сессию, т. е. только при первом заходе;
- **Never** — никогда. Страница всегда будет загружаться из кэша, а для обновления нужно нажать кнопку **Reload** (Перезагрузить) на панели инструментов браузера.

При работе через прокси-сервер или с использованием локального кэширования браузера или другой программы вы должны учитывать, что загружаемые страницы могут содержать устаревшие данные. Для обновления нужно вручную нажать кнопку **Reload**.

ГЛАВА 10

Передача файлов

Вспоминаю времена, когда построение сети было делом дорогим, а Интернет — еще дороже, и для обмена файлами приходилось бегать с дискетами 3,5 или 5,25 дюймов. Если кто-либо застал те времена, то, вероятно, вспоминает их с ужасом. Дискеты постоянно портились, и данные периодически переставали читаться. Благо, если расстояние небольшое, и можно было повторить пробежку, но когда дистанция стаерская, — потеря очень сильно отражалась на эмоциональном состоянии.

До сих пор в некоторые компьютеры вставляют дисководы для 3,5 дюймовых дискет, потому что дешевой альтернативы нет. Но уже трудно представить себе офис без полноценной сети, и в фирмах, где мне приходилось настраивать серверы, все компьютеры 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 не должно быть. Это касается не только программ, но и компьютерного железа.

Через дискеты хакеры легко могут унести секретную информацию, получив физический доступ к компьютеру. Мне известна фирма, которая содержала локальную сеть, оторванную от внешнего мира, и считала себя неприступной. Но когда секретная информация утекла с помощью простых дискет через все охранные системы, потому что металлоискатели не срабатывают на 3,5 дюймовые дискеты, и были потеряны рынки сбыта, только после этого все компьютеры лишились 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 возможности оставляют желать лучшего.

10.1. Работа FTP-протокола

Как мы уже говорили в *гл. 6*, для использования FTP-протокола требуется две составляющие: клиент и сервер. Для работы с FTP-сервером можно использовать любой Telnet-клиент, и подключившись к 21 порту сервера, вручную передавать команды. Но во времена графического интерфейса нужно что-то более удобное. В ОС Windows мой любимый FTP-клиент — CyD FTP Client XP (**www.cydsoft.com**), главное окно программы можно увидеть на рис. 10.1.

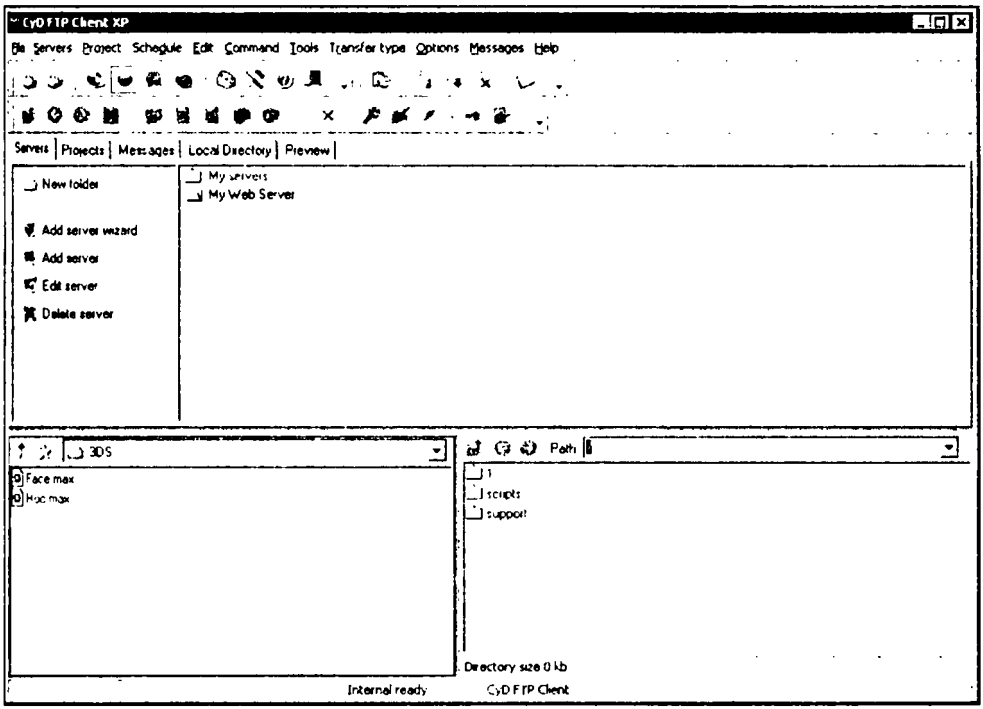

Рис. 10.1. Главное окно программы CyD FTP Client XP

Если у вас нет FTP-клиента, то для тестирования протокола можно использовать даже браузер, например, Internet Explorer или Natscape. Для этого в строке URL нужно набрать адрес: 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 используется для управления (пересылка команд), а другой — для передачи данных (файлов). Программа-клиент соединяется с 21 портом и начинает передавать команды. К этому порту подключаются 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 с точки зрения администрирования, точнее сказать, конфигурирования сетевого экрана.

Большинство команд, используемых в протоколе FTP, схожи с теми директивами, которые вы используете в Linux для управления файлами. Это связано с тем, что во время разработки протокола основной сетевой ОС являлась Unix-система. Это в наше время везде стоит Windows, а 20 лет назад все было иначе.

10.1.1. Команды FTP-протокола

Давайте рассмотрим листинг 10.1. В нем приведен пример, в котором клиентская программа обменивается командами с FTP-сервером. Если в начале строки стоит знак ">", то текст из нее был отправлен серверу, иначе — это ответ FTP-сервера на нашу команду.

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 21 портом. В этой строке чаще всего находится текстовое описание сервера, с которым произошло соединение, и его версия. В данном случае здесь вместо конкретного названия 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 и версия FTP-сервера и сообщение о приветствии. Ничего страшного не видите? А я вижу — хакеру достаточно подключиться на 21 порт, чтобы знать, с каким FTP-сервером он имеет дело.

Дальнейшие действия взломщика легко предсказываются. Я бы запустил поиск по всем базам уязвимостей на предмет наличия в них информации о дырах в данной версии сервиса wu-ftpd. Пусть администратор помолится, чтобы я не нашел нужных сплоитов для использования бреши, или дыра была незначительной и не позволила мне ничего сделать с его системой.

После чтения строки сообщения можно посылать команды FTP-серверу. Но ничего особого выполнить не удастся, пока вы не представитесь серверу. Для этого нужно выполнить FTP-команды: `USER` с параметром имя пользователя, а затем `PASS`, указав пароль.

Серверы FTP позволяют работать с тремя типами авторизации: действительная, гостевая и анонимная. 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 E-mail-адрес. Честно говоря, можно ввести и чужой E-mail, например, соседа или кого-нибудь другого. Это сервер проконтролировать не сможет. Некоторые серверы вообще не проверяют корректность адреса даже по простым шаблонам, и можно передавать что угодно.

На практике анонимный доступ обладает минимальными возможностями чтения файлов и директорий и используется только в открытых файловых архивах. Имя Anonymous, чаще всего, используют для публикации открытых документов для всеобщего доступа через FTP. Например, разработчики про-

грамм создают анонимные FTP-серверы для того, чтобы пользователи могли скачать с сервера последние версии программ или обновления уже существующих.

Реальный пользователь может путешествовать по всей файловой системе, и возможности ограничены только правами доступа используемой учетной записи.

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, но, в отличие от действительного, он работает только в своей директории. Например, если гостю назначен каталог **/home/robert**, то он сможет безнаказанно здесь работать с файлами и подкаталогами, но выше этой директории подняться невозможно. Вы можете определить любое имя в качестве гостевого.

Обратите внимание, что пароль в команде PASS передается абсолютно в открытом виде. Это серьезная проблема. В каждой главе, где мы рассматриваем какой-либо сервис, доводится сталкиваться с открытой передачей данных. Ну что поделаешь, если на заре рождения Интернета никто не думал о хакерах. Теперь приходится изобретать разные методы, чтобы спрятать пароль.

Если ваш сервер обслуживает только анонимные соединения, то пароли могут передаваться как угодно. При такой аутентификации любой пользователь итак может подключиться к серверу, указав произвольный E-mail-адрес в качестве пароля. Но подобные серверы используются только с общедоступными ресурсами/файлами. При наличии важной информации доступ происходит по паролю, и необходимо сделать так, чтобы он шифровался. Для этого можно использовать stunnel или уже готовый протокол SFTP, который мы рассматривали в *разд. 5.3.8*.

Отличное решение я видел на одном из публичных Web-серверов около 10 лет назад. Для того чтобы закачать данные на сервер, необходимо было зарегистрироваться, заполнив Web-форму с личными данными. Затем вам выдается пароль на доступ, который действует только в течение одной сессии. После этого пароль уничтожается, и повторное подключение становится невозможным. Файлы закачиваются в специальную директорию, в которую разрешена лишь запись. Самим файлам устанавливаются права только для чтения и записи, а выполнение остается недоступным. Таким образом, пароли можно передавать в открытом виде. Даже если хакер увидит его, подключиться не сможет.

Реализовать одноразовые пароли достаточно просто, если ваш сервер использует подключаемые модули аутентификации PAM *(см. разд. 3.3.3)*.

После того как вы авторизовались на сервере, можно выполнять любые другие команды FTP-сервера. Но тут есть одна проблема — список директив зависит от сервера. Конечно же, есть определенные требования, которых придерживаются все производители, это основные команды, которые описаны в RFC (Requests for Comments, рабочие предложения). Так как возможности, предоставляемые стандартом, уже устарели, то разработчики Web-серверов начали добавлять свои функции, которые могут отличаться в разных версиях программ. Так что, если клиентская программа в каких-то ситуациях ведет себя не совсем корректно, то это еще не значит, что она плохая, просто она может быть несовместима с данным сервером.

Основные команды FTP-протокола вы можете увидеть в табл. 10.1. При работе через Telnet и для тестирования сервера они вам могут пригодиться.

Таблица 10.1. *Команды FTP-протокола*

Команда	Описание
`USER имя`	Используется при авторизации для указания имени пользователя
`PASS пароль`	Предназначена для указания пароля при авторизации
`SYST`	Вернуть тип системы
`HELP`	Предоставить список доступных для выполнения команд
`LIST`	Вывести список файлов и каталогов текущей директории
`PWD`	Возвратить текущую директорию
`CWD директория`	Сменить текущую директорию
`TYPE тип`	Указать тип передачи данных: A для ASCII, I — для бинарных файлов
`RETR параметр`	Скачать с сервера файл, указанный в качестве параметра
`STOR параметр`	Загрузить на сервер файл, указанный в качестве параметра
`ABOR`	Прервать последнюю FTP-команду или передачу данных
`QUIT`	Выйти из системы

10.1.2. Сообщения сервера

В ответ на полученные команды сервер возвращает нам сообщения, по которым можно оценить результат работы. Уведомления состоят из числового трехзначного кода и необязательной текстовой части. Если для ответа требуется несколько строк, то в первой (после кода) стоит дефис, а в последней (после кода) — идет пробел.

Вы должны знать смысл числовых кодов, чтобы определить тип ошибки. Когда к вам за помощью обращается пользователь, зная значения кодов, можно определить причину сбоя и быстро решить проблему.

Значения первой и второй цифр кода ответа FTP-сервера можно увидеть в табл. 10.2 и 10.3 соответственно.

Таблица 10.2. *Значения первой цифры в коде ответа FTP-сервера*

Код	Описание
1	Команда удачно запустилась, но еще не закончила свое выполнение, поэтому нужно дождаться ее окончания перед продолжением работы. Такие сообщения приходят во время выполнения длительных операций (например, запрос на передачу файлов). После завершения работы команды будет получено еще одно сообщение
2	Команда выполнена удачно, можно продолжать работу и посылать новые директивы. Такие сообщения приходят в ответ на простые команды
3	Команда выполнена удачно, но для завершения работы требуется дополнительная директива. Такие сообщения можно увидеть в ответ на операции, предусматривающие несколько действий, например, во время аутентификации, которая требует двух команд. Сообщение с кодом, начинающимся с цифры 3, появляется между отправкой команд `USER` и `PASS`
4	Команда выполнена неверно, но результат может быть положительным, если повторить попытку позже. Сообщение может быть получено в случае, если сервер не в состоянии сейчас выполнить команду из-за выполнения другой операции
5	Команда выполнена неверно. Это самый критичный ответ, который может быть при неверном синтаксисе директивы или неправильном задании параметров

Таблица 10.3. *Значения второй цифры в коде ответа FTP-сервера*

Код	Описание
0	Синтаксическая ошибка, команда воспринята неверно
1	Информация
2	Установка соединения
3	Сообщение относиться к аутентификации
4	Не определено
5	Сообщение файловой системы

Рассмотрим пример. Допустим, что пользователь увидел в своей программе FTP-клиент следующее сообщение и не знает, что делать дальше:

```
331 Anonymous access allowed, send identity (e-mail name) as password.
```

Вам достаточно только знать код 331. По первому числу 3 видно, что директива выполнена удачно, но требуется дополнительная команда. Вторая цифра тоже 3, т. е. сообщение появилось в ответ на аутентификацию. Когда может быть такой отклик? Конечно же, после ввода имени. Сервер FTP ожидает пароль, поэтому выдал сообщение с кодом 331.

Как видите, минимальных знаний достаточно, чтобы понять, какая у пользователя или на сервере возникла проблема, и можно максимально быстро ее решить.

10.1.3. Передача файлов

Так как протокол FTP предназначен для работы с разными системами, то для передачи файлов используются два основных режима — текстовый (ASCII) и бинарный.

Допустим, что 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). В Windows то же самое обозначается двумя символами <CR> и <LF> (Line Feed, перевод строки, код 10). Если после передачи открыть в Windows этот текстовый файл, то все строки сольются в одну, потому что нет символа <LF>.

На рис. 10.2 показан файл sendmail.cf (используется для конфигурации сервиса Sendmail), скаченный 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> в виде квадрата.

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. 10.3 можно увидеть файл sendmail.cf, полученный с сервера в ASCII-режиме. Теперь информация стала читабельной, и все символы перехода на новую строку расставлены самой ОС верно, в соответствии со своими внутренними правилами.

Бинарные файлы (например, картинки или музыку) необходимо передавать в двоичном режиме. Здесь нет различий, в какой ОС создан объект, и он будет правильно воспринят в любой операционной системе, умеющей работать с данным форматом.

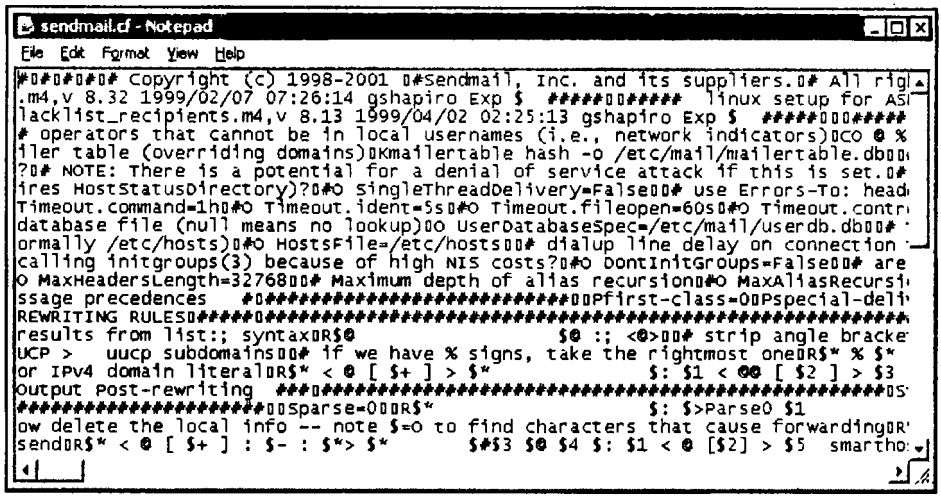

Рис. 10.2. Файл sendmail.cf, полученный с Linux-сервера в бинарном режиме

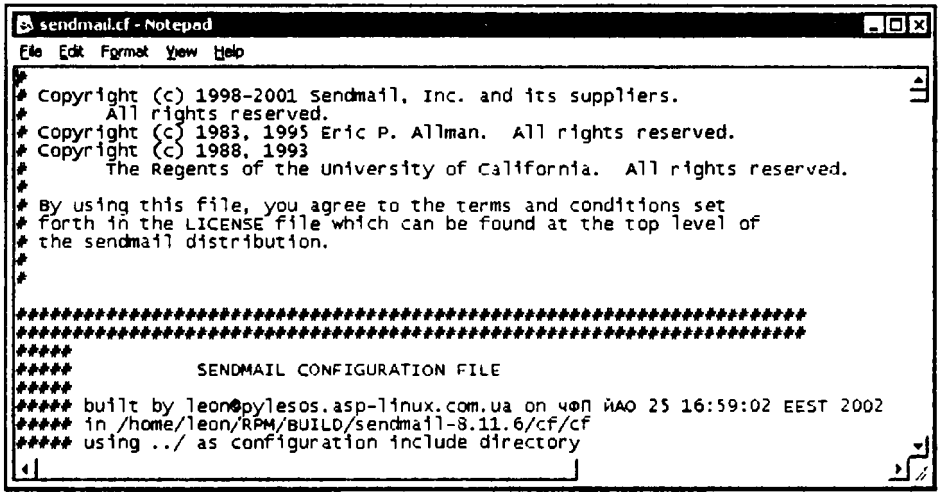

Рис. 10.3. Файл sendmail.cf, полученный с Linux-сервера в ASCII-режиме

Если передать двоичный файл из Linux в Windows в текстовом режиме, то любой символ <CR> (а он может встретиться и в двоичном файле) будет заменен на пару символов <CR> и <LF>, и после этого бинарный файл может стать нечитаемым.

10.1.4. Режим канала данных

Мы уже говорили о том, что для работы с FTP-сервером необходимо два канала. Порт 21 является командным, и по нему передаются только FTP-

команды. Для передачи файлов создается отдельное соединение. Этот процесс можно описать следующим образом:

- программа-клиент открывает порт на компьютере, куда сервер должен передать содержимое файла;
- серверу направляется запрос на скачивание файла и сообщается порт и IP-адрес клиентского компьютера, с которым он должен соединиться;
- сервер инициализирует соединение с компьютером клиента и начинает передачу данных.

Такой режим называется активным, потому что сервер устанавливает соединение. 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.

Таким образом, в активном режиме протокол FTP не будет работать через правильно настроенный сетевой экран. Если вы разрешите внешним программам устанавливать соединения, то можете отключить сетевой экран, 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 начинают устанавливать по умолчанию, потому что сетевые экраны в наше время уже встроены почти во все ОС.

В пассивном режиме соединение происходит иначе:

- клиент запрашивает скачивание или просит принять файл;
- сервер осуществляет привязку к порту и сообщает клиенту номер канала, куда необходимо подключиться для получения или отправки данных;
- 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 происходят только со стороны клиента. Это уже не противоречит правилам сетевого экрана.

10.2. Конфигурирование wu-ftp-сервера

По моим наблюдениям самым распространенным FTP-сервером является wu-ftp (Washington University FTP Server), потому что он поставляется вместе с основными дистрибутивами Linux, в том числе Red Hat и его клонами. Если у вас именно такой дистрибутив, то с установкой проблем не будет. Остается только правильно сконфигурировать сервис. Но даже если сейчас в системе

нет FTP-сервера, его легко установить из RPM-пакета (для Red hat-систем) или другого архива.

Для конфигурирования FTP в современных дистрибутивах есть графическая утилита kwuftpd. Для ее запуска выберите основное меню KDE, а затем нажмите **System/kwuftpd**. Главное окно программы вы можете увидеть на рис. 10.4. Но, как всегда, мы будем рассматривать более тонкую настройку через конфигурационные файлы. Разобравшись с этим, вы без проблем сможете настроить свой FTP-сервер и с помощью графической утилиты kwuftpd.

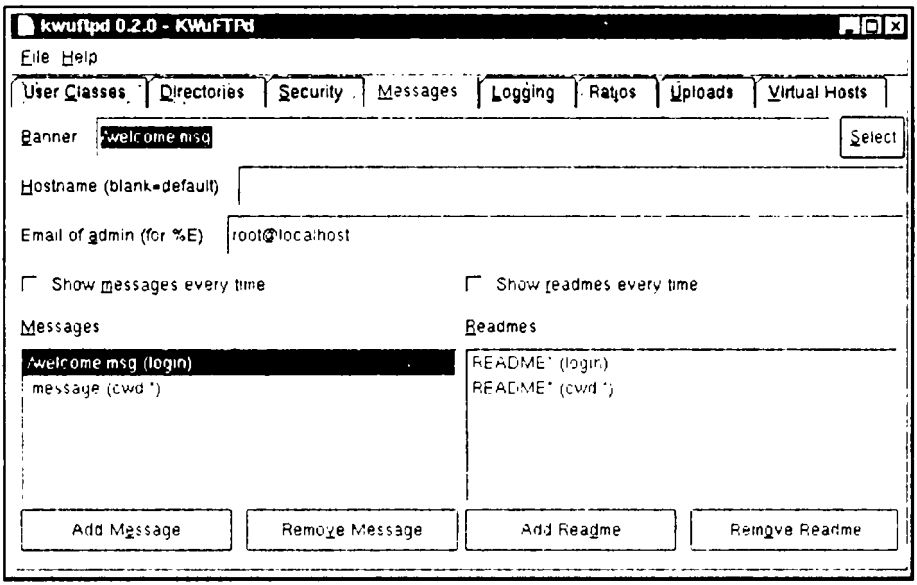

Рис. 10.4. Главное окно программы kwuftpd

Итак, конфигурация FTP-сервера содержится в шести файлах:

- ftpaccess — определяются права доступа к серверу, круг пользователей FTP и основные настройки безопасности;
- ftpservers — описываются виртуальные FTP-серверы;
- ftpusers — указываются пользователи, которым явно запрещен доступ к FTP-серверу;
- ftphosts — определяются права доступа к серверу с определенных хостов. Можно как запрещать, так и разрешать доступ;
- ftpgroups — описываются группы FTP;
- ftpconversion — дает нам возможность настройки преобразования файлов на лету.

10.3. Основные настройки wu-ftp-сервера

Основным конфигурационным файлом wu-ftp-сервера является ftpaccess. Его содержимое можно увидеть в листинге 10.2.

Листинг 10.2. Пример конфигурационного файла ftpaccess

```
# This file was generated by the KDE wu-ftpd configurator.
# (c) 2000 by Bernhard Rosenkrдnzer (bero@redhat.com)
class all anonymous,guest,real  *
noretrieve
loginfails 5
private no
banner /welcome.msg
email root@localhost
message/welcome.msg             LOGIN
message.message                 CWD=*
readme            README*       LOGIN
readme            README*       CWD=*
chmod               no          anonymous,guest
delete              no          anonymous
overwrite           no          anonymous
rename              no          anonymous
passwd-check        rfc822  warn
log transfers anonymous,guest,real inbound
log transfers anonymous,guest,real outbound
anonymous-root /home/flenov
```

При настройке FTP-сервера существует множество директив, для которых администраторы сохраняют исходные значения, потому что их работа не влияет на производительность или безопасность. Хотя некоторые из них помогают предотвратить неправильное или неэффективное использование сервера (например, время ожидания `timeout XXXX` способствует освобождению ресурсов), но значений по умолчанию достаточно, кроме того, изменение некоторых из них может, наоборот, сделать работу менее комфортной.

Давайте рассмотрим основные директивы этого файла. Чтобы впоследствии с ними легче было работать, я разбил все команды по категориям.

О дополнительных возможностях, которые мы не рассмотрим, вы можете получить подробную информацию из страниц руководства, выполнив команду `man ftpaccess`.

10.3.1. Доступ

Такие директивы определяют основные права доступа к FTP-серверу. Давайте рассмотрим основные из них:

- `class имя type адреса` — позволяет организовать классы пользователей по их типу и адресу. В нашем конфигурационном файле указана следующая строка:

  ```
  class all anonymous,guest,real *
  ```

  В качестве имени класса здесь указано `all`. После этого идет перечисление через запятую типов пользователей, которые будут отнесены к этому классу. В данном случае присутствуют все категории: `anonymous`, `guest` и `real` (анонимные, гости и авторизованные пользователи). Последний параметр — это шаблон адреса, на месте которого стоит звездочка, т. е. любой адрес. Получается, что к классу `all` относятся любые пользователи с произвольными адресами.

  Классы — очень удобная вещь. Это как группы. Вы объединяете определенных пользователей и можете назначать им права. Например, можно создать класс пользователей, у которых IP-адрес относится к зоне вашей компании, офиса или страны. Затем только этому классу открываете полноценный доступ к FTP, а всем остальным — запрещаете или ограничиваете использование. Назначить права сразу целому классу намного удобнее, чем расписывать разрешение каждому пользователю в отдельности;

- `noretrieve тип класс файл` — запрещает чтение указанного файла. Параметр `тип` указывает на абсолютную (absolute) или относительную (relative) адресацию (путь) к файлу. Далее идет определение класса в формате `class=имя класса`, к которому относится данный запрет. Можно явно указать описанный выше класс `all` или вообще его опустить, тогда запрет будет относиться ко всем пользователям. Если в качестве файла указан конкретный путь, то доступа не будет только к этому файлу. Если в параметре указано лишь его имя (например, passwd), то будет закрыт доступ ко всем файлам с таким названием в любой директории.

  Рассмотрим пример, запрещающий доступ к любым файлам с именем passwd:

  ```
  noretrieve relative passwd
  ```

  Попробуйте добавить эту строку в свой конфигурационный файл. Теперь подключитесь к серверу с помощью FTP-клиента. Для тестирования из X Window я в графическом режиме Linux использовал программу gftp. Подключившись к серверу, я создал файл passwd в директории **/home** и

попытался его скачать в каталог **/home/flenov**. В ответ на это FTP-клиент только создал пустой файл, но скачать ничего не смог, из-за установленного запрета программа завершилась аварийно. Такое окончание — это, конечно же, особенность gftp, а другой FTP-клиент должен правильно обработать ошибку и сохранить работоспособное состояние.

Если FTP-сервер находится на одном физическом сервере с Web-сервером, то вполне логичным будет запретить чтение файла **.htaccess**, в котором прописываются права доступа на директории для Web-сервера. Пользователи FTP не должны иметь права даже смотреть их. Лучше прописать разрешение только конкретным клиентам, чтобы каждый из них мог работать только со своими файлами **.htaccess**, или предоставить другой способ редактирования прав.

Работу с системными каталогами можно вообще отменить. Например, следующая строка запретит получение любого файла из директории **/etc**:

```
noretrieve /etc
```

После этого нельзя будет скачать ни один файл не только из директории **/etc**, но и из ее поддиректорий;

- `loginfials число` — количество неудачных попыток входа на сервер, после которого в журнале будет создана соответствующая запись. В нашем примере стоит число 5. Если пользователь не смог войти на сервер 5 раз, то это уже говорит о попытке взломщиком подобрать пароль случайным образом или по словарю (атака brute force);
- `private параметр` — возможность использовать команды wu-ftp-сервера `SITE GROUP` и `SITE GPASS` (в других FTP-серверах этих команд может и не быть) для смены группы (значение параметра `yes`). Если пользователь укажет верную группу и пароль, то он получит права группы из файла ftpgroups;
- `deny адрес файл` — запрет доступа клиентов с указанного адреса. В случае попытки подключения выводится сообщение из текстового файла, указанного в качестве последнего параметра. Адрес может указываться в виде шаблона;
- `defumask mask` — маска прав доступа, используемая при создании новых файлов. О команде `umask` для ОС Linux (задает текущее значение маски) мы говорили в *разд. 4.1*;
- `limit-time тип минуты` — ограничение времени сессии. Например, вы хотите, чтобы определенные пользователи не засиживались на вашем FTP-сервере. Используя эту директиву, в качестве типа можно указать звездочку (`*`) для всех пользователей, `real`, `anonymous` или `guest`. Последний пара-

метр — это количество минут сессии. По прошествии указанного времени соединение будет разорвано;

- `file-limit направление число класс` — ограничение на число передаваемых файлов. В качестве направления можно указать `in` (входящие), `out` (исходящие) и `total` (всего). Чтобы запретить работу более чем с 10 файлами, воспользуйтесь командой `file-limit total 10`;
- `byte-limit направление число класс` — ограничение на число передаваемых байт. Работа директивы схожа с `file-limit`;
- `anonymous-root каталог` — задание в явном виде корневой директории для анонимных пользователей, т. к. у них не может быть собственного каталога в отличие от реального пользователя, для которого при подключении к системе корнем является его домашняя директория;
- `guest-root каталог` — аналогична предыдущей команде. Директива необходима, если вы хотите, чтобы все гости могли работать с одной и той же директорией. Если у каждого гостя должен быть свой каталог, то лучше для каждого из них явно создавать учетную запись *(см. разд. 10.6)*.
- `passwd-check тип сообщение` — определяет проверку правильности пароля для анонимных пользователей. В данном случае имеется в виду контроль E-mail-адреса, который они используют. В качестве типа может указываться одно из трех значений: `none` (нет проверки), `trivial` (простая проверка на содержания в адресе символа "@)" или `rfc882` (полная проверка, на соответствие стандарту rfc 822). Параметру `сообщение` можно присваивать значение `warn` (выводить предупреждение, но продолжать работу) или `enforce` (отказать в доступе);
- `deny-email адрес` — отказ в доступе, если в качестве пароля используется указанный адрес. В большинстве FTP-клиентов в настройках прописан для анонимного доступа какой-либо почтовый ящик, например, **my@mail.com**. Мало кто меняет этот адрес. Так как он соответствует всем правилам, то сервер не определит, что это обманка. Но можно прописать его в этом параметре, и тогда пользователю придется поменять в настройках FTP-клиента адрес на другой, иначе он не сможет подключиться. Но даже это не будет гарантировать, что анонимный пользователь указал именно свой E-mail в качестве пароля;
- `deny-uid идентификаторы` — запрещает доступ к FTP пользователям с указанными идентификаторами. Те же самые функции выполняет файл ftpusers, который мы рассмотрим в *разд. 10.5*. Удобство этой команды в том, что можно задать диапазоны. Например, `deny-uid %-500`. Данная директива запретит доступ всем пользователям, у которых идентификатор менее 500;

- `deny-gid идентификаторы` — запрещает доступ к FTP пользователям группы, с указанными идентификаторами. Те же самые функции выполняет файл ftpusers;
- `restricted_uid идентификаторы` — разрешает гостевому пользователю с указанным ID получать доступ к директориям вне его домашнего каталога;
- `restricted_gid идентификаторы` — дает право группе пользователей с указанным ID получать доступ к директориям вне домашнего каталога;
- `unrestricted_uid идентификаторы` — запрещает гостевому пользователю с указанным ID получать доступ к директориям вне его домашнего каталога;
- `unrestricted_gid идентификаторы` — запрещает группе пользователей с указанным ID получать доступ к директориям вне домашнего каталога;
- `dns refuse_no_reverse файл override` — выдать сообщение, если клиент не имеет обратного адреса. При отсутствии параметра `override` соединение будет завершено;
- `dns refuse_mismatch файл override` — выдать сообщение, если прямой и обратный адреса не совпадают. Если не указать параметр `override`, то соединение будет завершено. По умолчанию я всегда включаю эту опцию, а отключаю, только если у действительных пользователей возникают проблемы при работе с сервером. Это необходимо для того, чтобы взломщик не мог подделать IP-адрес для входа в систему и обхода соответствующей проверки.

10.3.2. Контроль загрузки файлов

Загрузка файлов — самая опасная возможность для сервера. Каждый пользователь должен иметь право обращаться только к своей директории. А что делать, чтобы анонимные пользователи тоже могли работать с файлами? В этом случае нужно по возможности запретить загрузку в уязвимые с точки зрения безопасности директории, куда злоумышленник может поместить скрипты и выполнить их.

`upload параметры` — команда позволяет настроить права на загрузку файлов в определенные каталоги. Для запрета загрузки в каталог **/etc** нужно написать следующую строку:

```
upload /etc    no
```

Давайте посмотрим, как можно разрешить загрузку в директорию:

```
upload /home     yes root root 0600 nodir
```

В данной строке разрешается загрузка файлов в директорию **/home**. Третий и четвертый параметры определяют владельца и группу, которые будут установлены для файла. В обоих случаях я указал root, чтобы простой смертный после загрузки ничего не смог сделать с документом. Далее идут права доступа на файл — `0600`, т. е. читать и писать в него сможет только администратор, а все остальные отдыхают. Последний параметр равен `nodir`, что соответствует запрету на создание директорий.

Следующий пример разрешает создавать директории:

```
upload /home/robert  yes root root 0600 dir 0700
```

Предпоследний параметр равен `dir`, что позволяет создавать директории. Последний параметр `0700` — это права доступа на каталог (только администратору разрешено все). Изменение этого значения позволяет сделать возможной работу с файлами только администратору root. Даже если злоумышленник загрузит в указанную директорию свою программу, выполнить он ее не сможет, не имея соответствующих прав.

Запретите загрузку файлов в любые системные директории, открытые пользователям для просмотра. Если вы используете исключительно гостевой доступ, когда пользователь может работать только в своем окружении, то в этом уже нет крайней необходимости.

10.3.3. Доступ по операциям

Помимо этого, в файле ftpaccess может быть описание основных операций и прав доступа к ним. В общем виде это выглядит следующим образом:

```
Действие yes|no пользователь
```

В качестве действия можно указать одно из следующих значений: `chmod`, `delete`, `overwrite`, `rename` или `umask`. Затем идет разрешение (`yes`) или запрет (`no`). Потом через запятую могут перечисляться типы пользователей (`anonymous`, `guest` или `real`) или указывается класс (`class=имя_класса`).

По умолчанию все действия и для всех пользователей разрешены. Но вполне логичным будет запретить удаление, изменение атрибутов, переименование или перезапись файлов неавторизованным (anonymous) пользователям.

Например, в листинге 10.2 для запрещения доступа к операциям есть следующие строки:

```
chmod     no     anonymous,guest
delete    no     anonymous
overwrite no     anonymous
rename    no     anonymous
```

10.3.4. Информационные директивы

Эти директивы отвечают за информационные сообщения, которые видит пользователь, работая с вашим FTP-сервером:

- `banner имя` — в качестве имени можно указать текстовый файл, содержимое которого будет передано пользователю во время входа в систему. Этот файл может содержать приветствие, полезную информацию или правила использования вашего FTP-сервера. Вы должны помнить, что баннер виден пользователю еще до авторизации, поэтому здесь не должно быть никаких сведений, которые помогут хакеру при взломе;
- `greeting параметр` — определяет, какую информацию выдавать пользователю о системе после отображения баннера. Это текстовая строка, которая может выглядить следующим образом: "220 flenovm.ru FTP server (Version wu-2.6.2-5) ready.". Как мы уже говорили ранее *(см. разд. 10.1)*, эта строка отображается до авторизации, и в ней содержится информация о системе и версии используемого FTP-сервера. Это лишнее, злоумышленник не должен видеть этого, лучше выводить минимум полезного или даже что-то запутывающее. В качестве параметра можно использовать одно из следующих значений:
  - `full` — полная информация, включая имена хоста и программы и версия;
  - `brief` — отображать сокращенную информацию, включающую только имя хоста;
  - `terse` — только сообщение о готовности системы к работе;
  - `text` — собственное сообщение.

  Больше всего мне нравится последний параметр. При его использовании через пробел нужно указать текст, который будет отображен пользователю, например:

```
greeting text строка
```

  На собственных серверах я использую сообщения именно этого типа (одно из двух):

```
greeting text flenovm.ru FTP Server (MS IIS 4.1.0) ready
```

  или

```
greeting text flenovm.ru FTP Server (cd-ftpd 2.1.9) ready
```

  Увидев любое из этих сообщений, хакер не сможет определить, какой FTP-сервер реально установлен. В первом случае он подумает, что имеет дело с IIS (Internet Information Services, информационные сервисы Интер-

нета) разработки Microsoft, который может работать только в среде Windows. Это может сбить с толку любого, даже опытного хакера. Но, проверив сервер специальными утилитами, он увидит, что реально используется Linux (точную версию такие утилиты, чаще всего, не могут определить) и догадается, что его просто кидают.

Во втором случае я вообще подставляю имя несуществующего сервера. Взломщик, не зная такого сервера и не найдя нужной программы для проникновения, будет искать другие пути или вообще может уйти восвояси.

Можно притвориться, что у вас установлен сервер ProFTP, который реально существует и тоже достаточно часто используется администраторами Linux;

- `hostname имя` — параметр определяет имя локального хоста по умолчанию;
- `email адрес` — это адрес электронной почты администратора;
- `message файл тип` — содержимое указанного текстового файла, отображается пользователю в следующих случаях:
  - при входе в систему, если в качестве типа задано слово `LOGIN`;
  - при смене директории, если в качестве типа указано `CWD=каталог` и пользователь вошел в указанный каталог.

10.3.5. Журналирование

Некоторые администраторы фиксируют только действия анонимных и гостевых пользователей. Это обосновывается тем, что реальные пользователи системы не будут портить сервер, который им нужен для работы или обмена файлами. В корне неверное мышление, потому что достаточно много взломов совершается именно real-пользователями, да и злоумышленники, чаще всего, используют их.

Навредить серверу через гостевую или анонимную учетную запись слишком сложно, а при правильном распределении прав невозможно (только если есть ошибка в программе сервера). В большинстве случаев злоумышленник стремится заполучить доступ к авторизованной учетной записи и использовать ее.

Если произошло что-то неординарное, то, используя журналы, вы должны получить максимальную информацию о происшедшем и сделать все необходимое, чтобы это не повторилось (к вопросу журналирования мы еще вернемся в *гл. 12*).

По умолчанию wu-ftp для хранения журнала, который содержит последние записи FTP-сервера, использует файл **/var/log/xferlog**. История за предыду-

щие дни может быть просмотрена в файлах **/var/log/xferlog.X**, где X — это число от 0 до 5.

Давайте посмотрим, что можно регистрировать в журналах wu-ftp-сервера:

- `log commands список` — сохранять все команды клиента. В качестве списка можно указать одно из следующих значений: `anonymous`, `guest` или `real`;
- `log transfers список направление` — писать файлы, которые получает или отправляет пользователь. Последний параметр может содержать одно из двух значений `inboard` (входящий) и `outboard` (исходящий);
- `log security список` — регистрировать все нарушения безопасности, попытки выполнить запрещенные команды и др.;
- `log syslog` — использовать системный журнал syslog;
- `log syslog+xferlog` — использовать журналы syslog и xferlog одновременно.

10.4. Создание виртуальных серверов

Поддержка виртуальных серверов для FTP — очень мощная вещь. Когда на компьютере крутится 20 виртуальных Web-серверов и ими управляют разные люди, то вполне логичным будет для каждого из них поставить в соответствие свой FTP-сервер. В этом случае любому сайту могут быть назначены свои правила.

Мы не будем сильно заострять внимание на виртуальных серверах, потому что это выходит за рамки нашей книги. Если вы хотите узнать о их параметрах, то можете обратиться к документации (`man ftpaccess`), или для создания воспользоваться графической утилитой управления FTP-сервером.

Если честно, то мне не очень нравятся возможности wu-ftp по работе с виртуальными серверами. Если вам действительно нужно несколько серверов на одном компьютере, то советую обратить внимание на сервер ProFTP, который, на мой взгляд, больше приспособлен для решения данной задачи. Неудобство настройки wu-ftp заключается в разбросе конфигурационных файлов.

Определение всех виртуальных FTP-серверов происходит в конфигурационном файле ftpservers. Описание выполнено в виде строк, каждая из них содержит IP-адрес виртуального сервера и каталог, в котором находятся его файлы настройки (дублирующие все указанные ранее файлы wu-ftp-сервера). Если один из них отсутствует, то будет использоваться соответствующий конфигурационный файл из каталога **/etc**.

10.5. Дополнительные настройки

Пока что мы рассматривали только файл ftpaccess. Но мы уже знаем, что для конфигурации wu-ftp-сервера используется намного больше файлов. Давайте посмотрим, что еще нам доступно для настроек.

10.5.1. Запрет доступа реальным пользователям

Так как сервер wu-ftp работает с учетными записями ОС, которые расположены в файле **/etc/passwd**, то любой пользователь автоматически сможет работать с FTP-сервером, используя свой аккаунт и права доступа. Но далеко не всем это необходимо.

Чтобы запретить доступ уже существующему пользователю, его имя нужно добавить в файл **/etc/ftpusers**. Содержимое этого файла по умолчанию можно увидеть в листинге 10.3. В зависимости от дистрибутива текст может изменяться.

Листинг 10.3. Содержимое файла /etc/ftpusers

```
# The ftpusers file is deprecated.
# Use deny-uid/deny-gid in ftpaccess.
root
bin
daemon
adm
lp
sync
shutdown
halt
mail
news
uucp
operator
games
nobody
```

Обратите внимание, что пользователю root тоже запрещен доступ. Это связано с тем, что у администратора слишком много прав, и если злоумышленник получит возможность пользоваться этой учетной записью, то сможет удаленно уничтожить систему (например, стереть файлы). Никогда не допускайте к FTP пользователей с повышенными привилегиями (администратора root и пользователей группы root).

Если вам нужно работать с файлами или директориями, которые доступны только администратору с нулевым идентификатором, то не используйте для этого FTP. Лучше производить изменения, сидя непосредственно за компьютером или через промежуточную папку (закачать файлы в свой каталог, а потом локально или через удаленный, но безопасный терминал, корректировать).

Лучше всего доступ по FTP запретить всем системным учетным записям, ID которых менее 500. Для этого достаточно в файле ftpaccess добавить следующую строку:

```
deny-uid %-500
```

В этом случае можно быть уверенным, что никого не забыли. Тем более что с одним идентификатором (например, с ID, равным нулю) может быть несколько пользователей.

10.5.2. Компьютерам вход запрещен

Как говорит великая администраторская мудрость — на сетевой экран надейся, а сам не плошай. Firewall позволяет запретить доступ к серверу на определенные порты с конкретных компьютеров. Конфигурационный файл **/etc/ftphosts** выполняет схожие задачи — разрешает или запрещает доступ с указанных IP-адресов или целой сети.

По умолчанию файл пустой, потому что разработчики дистрибутива не могут знать, как вы планируете организовать доступ. Вы можете прописать в нем следующие директивы:

```
allow имя шаблон
deny имя шаблон
```

Например, если вы хотите запретить доступ анонимным пользователям с адреса 192.168.1.1, то добавьте строку:

```
deny anonymous 192.168.1.1
```

Если исходить из нашего принципа "Что не разрешено, то запрещено", то может показаться, что строка с директивой `deny` не нужна. Это неверно, потому что необходимо закрепить разрешение на доступ с указанного адреса для определенного типа пользователей, чтобы остальным закрыть возможность входа на FTP.

10.5.3. Группировка

В файле ftpgroups находятся описания групп (равносильны правам доступа), при создании которых можно использовать команды SITE GROUP и SITE

GPASS. Это нестандартные директивы FTP, и мало кем из производителей поддерживаются, поэтому для пользователей работа с этими командами может оказаться слишком неудобной.

Файл ftpgroups содержит строки примерно следующего вида:

```
test:ENCRYPTED PASSWORD HERE:archive
```

Строка-описание состоит из трех параметров, разделенных двоеточием: имя группы, пароль и реальное (системное) имя группы.

10.6. Гостевые учетные записи

Если сейчас попытаться войти в систему под любым аккаунтом, то вы сможете путешествовать по всей файловой системе. Но в большинстве случаев пользователям нужно работать только с собственными документами, поэтому для всех своих FTP-пользователей я завожу гостевые учетные записи. Давайте рассмотрим на примере, как это происходит.

Для начала нужно создать новую учетную запись для пользователя, например robert_ftp. Для этого выполняем команду:

```
add robert_ftp
```

Теперь посмотрим на созданную для него строку в файле **/etc/passwd**. Она должна выглядеть примерно следующим образом:

```
robert_ftp:x:507:507::/home/robert_ftp:/bin/bash
```

Классическая запись для нового пользователя. Но через нее можно входить в систему локально, а мы должны ограничиться только FTP-доступом. Изменим командную оболочку (shell) для этого пользователя на **/bin/ftponly**. Такой оболочки пока нет, и мы ее еще создадим чуть позже. Помимо этого, необходимо директорию **/home/robert_ftp** сделать корневой. Для этого нужно добавить в конце пути папку с именем в виде точки.

Отредактируйте соответствующим образом строку, и вы получите такой результат:

```
robert_ftp:x:507:507::/home/robert_ftp/.:/bin/ftponly
```

Обратите внимание, что в качестве командной оболочки указан файл **/bin/ftponly**, который не существует. Давайте его создадим. Это делается только один раз и потом используется всеми гостевыми учетными записями. Для создания файла можно воспользоваться командой `cat`:

```
cat >> /bin/ftponly
```

В ответ на это все команды, которые теперь будут вводиться в консоли, попадут в файл **/bin/ftponly**. Наберите следующий текст:

```
#! /bin/sh
echo 'You are not allowed to log in interactively'
exit 0
```

Для завершения ввода в файл нужно нажать сочетание клавиш <Ctrl>+<X>, и вы вернетесь в нормальный режим работы.

В файле **/bin/ftponly** у нас хранится всего две команды: первая выводит на экран эхо-сообщение о том, что нельзя входить в систему интерактивно, вторая — завершает сеанс.

Теперь необходимо сделать наш скрипт **/bin/ftponly** исполняемым. Для этого выполните команду:

```
chmod 755 /bin/ftponly
```

Итак, у нас создан файл командной оболочки и пользователь, который использует ее. Если сейчас попытаться войти в систему как пользователь robert_ftp, то на секунду появится сообщение "You are not allowed to log in interactively" и произойдет выход из системы. Таким образом, работать под учетной записью robert_ftp нельзя.

Вместо файла **/bin/ftponly** можно использовать в качестве командного интерпретатора **/dev/null** — нулевое устройство, которое не может обрабатывать команды и не позволит входить в систему под этой учетной записью. Оно определено в качестве консоли в файле **/etc/passwd** для всех системных учетных записей, которые не предназначены для локальной работы.

Теперь осталась самая малость — сказать серверу FTP, что пользователь с именем robert_ftp является гостем. Для этого в файл ftpaccess добавим строку:

```
guestuser robert_ftp
```

Если теперь подключиться к серверу по FTP, то пользователь сможет увидеть только свой каталог, который будет казаться корнем. Все остальные папки, расположенные выше, будут не видны.

В моих системах все пользователи работают только как гости, со своими каталогами или анонимно с общедоступными папками. Действительные учетные записи устанавливаются только избранным администраторам и только в крайних случаях, потому что ими сложнее управлять.

Для гостевых пользователей достаточно только ограничить доступ определенной директорией, а все остальное защитит сервер. Правда, и здесь иногда бывают проблемы. Рассмотрим классическую ошибку программистов. Допустим, что пользователю открыта директория **/home/robert**, и для обеспечения этого сервер банально проверяет, чтобы путь к каталогу начинался с этой строки. Хакер видит эту директорию как корневую(/) и выше подниматься не должен. Теперь посмотрите на следующую команду:

```
cat /home/robert/../../../../../etc/passwd
```

Она должна выводить на экран файл **/home/robert/../../../../../etc/passwd**, а на экране вы увидите содержимое **/etc/passwd**. С точки зрения проверки начала пути все записано верно, но после **/home/robert** идет множество символов "/..", каждый из которых заставляет передвинуться на уровень выше. Хакер, не зная в какой папке он находится, может поставить такую комбинацию раз пять и, скорее всего, поднимется до корня, а потом переместится в системную папку **/etc/passwd**.

Несмотря на эту простоту, ошибка встречается довольно часто. А ведь программисту достаточно проверить адрес на вхождение в него символов двоеточия, и если они есть, то принять меры. В wu-ftp-сервере такой ошибки нет, но нет гарантии, что она не появится в будущем, когда выйдет обновление, в котором по случайности кто-то отключит проверку или удалит ее. Поверьте мне, такое иногда бывает с любым программным обеспечением, особенно если над ним работает целая команда, и нет контроля.

10.7. Безопасность FTP-сервера

Пока что мы говорили о конфигурации FTP-сервера под Linux. А сейчас нам предстоит разобраться с некоторыми практическими примерами использования сервера не по назначению и методами защиты.

Примеры, которые мы будем рассматривать в этом разделе, повергли в шок сетевую общественность и специалистов по безопасности, потому что FTP-сервер можно использовать для реализации практически любой атаки: рассылка вирусов, троянских программ и спама, взлом сервера и даже анонимное сканирование портов удаленного компьютера. Сервер FTP может выступать посредником в работе хакера.

10.7.1. Перехват соединения

Вспомним полный цикл соединения с FTP-сервером и передачи файлов:

1. Соединение с сервером.
2. Авторизация.
3. Запрос на скачивание файлов.
4. Сервер открывает порт и сообщает его клиенту.
5. Клиент подключается к указанному порту и получает или передает файл.

Очень сложно, но вполне вероятно направить соединение на себя. Хакеру нужно перехватить пакет, в котором сервер сообщает порт, и раньше авторизованного клиента успеть подключиться к этому каналу и передать на сервер свой или получить чужой файл.

Самое страшное — именно передача файла. Так как хакер вклинивается уже после авторизации, то он без проблем может транслировать данные, а сервер не контролирует, с какого адреса было запрошено соединение, и какой IP реально подключился. Если хакеру удастся перехватить такое соединение, то он сможет закачать свою программу, содержащую вирус или, например, троянскую программу.

В настоящее время в большинство FTP-клиентов уже встроили сравнение IP-адресов, подключенных к 21 порту и каналу для передачи данных. Это усложняет атаку, потому что теперь хакеру необходимо подделывать IP-адрес, что в случае с TCP-протоколом не так уж и просто.

Использование привязки к IP не всегда позволяет решить проблему. Если на пути соединения с FTP-сервером находится анонимный прокси-сервер или сетевой экран, маскирующий IP-адрес, то сервер будет видеть не реальный IP-адрес клиента, а IP-адрес proxy или Firewall.

Можно запретить пассивный режим, что закроет данный вопрос полностью. Но это не является панацеей от всех проблем с безопасностью. В следующем разделе мы увидим, что активный режим также далеко небезопасен.

А чего ждать? Активное соединение тоже можно перехватить, хотя это сделать немного сложнее. Если хакер получил доступ к компьютеру, который подключен к FTP-серверу, то достаточно дождаться момента, когда пользователь взломанного компьютера запросит передачу данных, и перехватить порт.

10.7.2. Сканирование портов

В *разд. 1.1* мы говорили о том, что на начальном этапе взлома компьютера хакер должен получить как можно больше сведений о жертве. Средством сбора первичной информации является сканирование портов. Делать это с собственного компьютера опасно, поэтому хакеры стараются маскироваться и идут на различные хитрости.

Одна из уловок — использование сценария на PHP или Perl, который с сервера будет производить сканирование. Но этот метод обладает следующими недостатками:

- необходим сервер, который сможет выполнять сценарии, а не всегда есть возможность найти подходящий;
- бесплатные серверы с возможностью выполнения сценариев требуют регистрации и обладают хорошей системой журналирования. Если первое препятствие можно обойти, указав неверные данные, то второе является большой проблемой. Сервер зафиксирует сканирование, и служба безопасности может быстро найти будущего взломщика.

Хакеры нашли отличный способ заставить сервер сканировать порты. Для этого всего лишь нужно подключение к FTP-серверу, который работает в активном режиме.

Вспомним, как происходит прием/передача файлов в активном режиме. Для этого FTP-серверу посылается запрос с указанием порта клиентского компьютера, к которому он должен подключиться. Помимо номера порта передается и IP-адрес. Это значит, что клиент с адреса 192.168.1.1 может запросить соединение с компьютером 192.168.8.2 и любым его портом, и сервером это воспримется нормально. Хакеры научились использовать эту особенность для того, чтобы заставить FTP-сервер сканировать порты удаленного компьютера.

Таким образом мне удалось один раз удачно произвести атаку DoS на свой сервер. Я заставил FTP-сервис сканировать компьютер с proxy, через который происходило подключение к Интернету. На прокси-сервере была установлена система обнаружения атак, которая при выявлении попытки сканирования автоматически блокировала любые соединения с компьютером (о таких системах мы поговорим в *гл. 12*). Сканирование прошло удачно, и я со спокойной душой пошел на обед. Вернувшись, услышал от пользователей, что невозможна работа по FTP. Я проверил, сервер работал исправно. Оказалось, что FTP стал недоступен для внешних пользователей, подключающихся через прокси-сервер, который во время сканирования добавил FTP-сервер в черный список.

Для сканирования через FTP-сервер можно использовать программу nmap:

```
nmap -b имя_пользователя:пароль@ftpсервер:порт
```

Как видите, эта запись очень сильно похожа на строку подключения через Web-браузер. Если для сканирования будет использоваться анонимный сервер, то имя и пароль указывать не обязательно:

```
nmap -b ftpсервер:порт
```

Если сервер работает на 21 порту, то можно опустить и этот параметр.

Для защиты можно исключить активный режим FTP-сервера с помощью сетевого экрана, т. е. блокировать работу с 20 портом, который чаще всего используется для передачи файлов. В этом случае все соединения инициализируются только со стороны клиента.

10.7.3. Рассылка файлов

С помощью FTP-сервера можно рассылать E-mail-сообщения. Для этого создайте на сервере текстовый файл со следующим содержимым:

```
HALO mailserver.com
MAIL FROM: name@server.com
RCPT TO: recipient@server.com
DATA
Текст письма
.
```

Рассмотрим, что означают эти строки, которые на самом деле являются командами SMTP-сервера:

- `HALO mailserver.com` — строка приветствия SMTP-сервера. Параметр `mailserver.com` необходимо заменить на реальное имя сервера;
- `MAIL FROM: name@server.com` — адрес отправителя письма;
- `RCPT TO: recipient@server.com` — получатель письма;
- `DATA` — после команды идет содержимое письма.

Последняя строка в файле должна состоять только из одной точки, потому что SMTP-сервер символы <CR> и <LF> (конец строки и перевод каретки) воспринимает как конец письма. Когда вы нажимаете клавишу <Enter>, то в ОС Windows создается как раз такая пара символов. ОС Linux ограничивается только переводом каретки. Для нас главное, чтобы в тексте был переход на новую строку, а какой он, — не имеет значения, потому что файл будет передаваться в ASCII-режиме.

Этот файл загружаем на FTP-сервер и выполняем следующие две команды:

```
PORT 192,168,1,1,25
RETR filename
```

В первой строке мы просим сервер соединиться с компьютером, имеющим адрес 192.168.1.1. Для этого используется FTP-команда `PORT`, которой необходимо передать 5 чисел: первые 4 — это IP-адрес компьютера, последний параметр определяет порт. С помощью этой директивы можно производить сканирование портов на сервере вручную, но в данном случае у нас другая цель.

Вторая команда посылает на этот сервер файл с именем filename, который содержит SMTP-команды. Для SMTP-сервера все выглядит так, как будто FTP-сервер направляет ему директивы для отправки письма, которое он и отошлет. А получатель не сможет определить источник. Все параметры приведут его только к FTP-серверу. Получается, что злоумышленник может абсолютно анонимно отправить письмо, и его никто не найдет.

Таким способом могут рассылаться вирусы, троянские программы или даже спам. Есть еще один вариант использования почтовых сообщений — поместить на FTP-сервер большой файл и заставить бесконечно передавать его на

SMTP-сервер. Если запустить несколько таких процессов, то при слабеньком канале SMTP-сервера получится полноценная DoS-атака.

Защититься от этой атаки на стороне SMTP-сервера можно только с введением обязательной авторизации. В этом случае хакеру необходимо будет знать реальные параметры учетной записи, которой разрешен доступ. На стороне FTP-сервера также защищаемся с помощью аутентификации. Никаких анонимных подключений быть не должно, тем более с правами на запись файлов.

10.8. Дополнительная информация

Я не стал описывать все директивы конфигурационных файлов, которые доступны в wu-ftp-сервере. Их слишком много, и мы остановились только на основных.

Для получения дополнительной информации можно выполнить команду

```
man файл.
```

В качестве параметра укажите имя конфигурационного файла, о котором вы хотите узнать больше.

Помимо этого, можно почитать документы из директории **/usr/share/doc /wu-ftpd-X.X.X**, где X.X.X — это номер версии установленного у вас wu-ftp-сервера.

Все изменения, которые вносятся в конфигурационный файл, вступают в силу немедленно. Единственное ограничение — клиенты должны заново подключиться к серверу, иначе они будут продолжать работать со старыми настройками.

При администрировании FTP-сервера вам помогут следующие команды:

- `ftpd` — позволяет запустить сервер с особыми параметрами. Атрибутов очень много, поэтому для получения подробной информации о них можно обратиться к документации (выполнить команду `man ftpd`). Лично мне еще ни разу не приходилось прибегать к использованию ключей, потому что единожды настроив конфигурацию, сервер работает стабильно;
- `ftprestart` — используется для перезапуска FTP-сервера;
- `ftpshut` — корректно завершает работу сервера. Например, если вы хотите обновить программное обеспечение, то не стоит отключать сервис аварийно. Используйте эту команду со следующими ключами:
  - `-l n` — не принимать новые соединения за `n` минут до завершения работы FTP-сервера. Укажите приемлемое время, чтобы клиенты успели корректно завершить работу с сервером;

- `-d n` — разорвать соединения за `n` минут до завершения работы FTP-сервера. *Я рекомендую установить разрыв непосредственно перед завершением или указать 1 минуту;*
- `время` — задает момент завершения работы FTP-сервера и схож с аналогичным параметром в команде `shutdown` для Linux. Вы можете указать вместо времени ключевое слово `now`, чтобы завершить работу немедленно, но я рекомендую использовать ключ `+n` (где `n` — это количество минут до завершения работы) или указать точное время в формате ННММ (часы— минуты);

- `ftpcount` — выводит количество подключенных по FTP пользователей. Когда в системе происходит что-то неладное, то я всегда проверяю, есть ли соединения клиентов по FTP. Если да, то следующим этапом нужно узнать, кто подключен;
- `ftpwho` — возвращает список подключенных к FTP-серверу клиентов с указанием учетной записи, использовавшейся при подключении. Иногда одного взгляда достаточно, чтобы определить соединение злоумышленника. Например, имеется учетная запись человека, который не может быть в данное время на FTP;
- `ckconfig` — проверяет конфигурацию FTP-сервера и выводит отчет для каждого конфигурационного файла wu-ftp-сервера.

10.9. Резюме

Протокол FTP и серверы различных производителей за время своего существования имели множество проблем с безопасностью. Если сложить ошибки 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 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 HTTP-протокол. Он безопаснее, и 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-протокол. Многие администраторы не хо-

тят настраивать Web-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 Web-сервер, то максимально используйте его возможности, и тогда можно будет закрыть 21 порт, тем самым оградив себя от вероятных ошибок, которые могут с ним прийти.

Если вам лично необходим доступ к серверу для работы с файлами удаленно, то советую использовать пакет SSH и встроенный SFTP-протокол, который шифрует данные, такое соединение перехватить намного сложнее.

ГЛАВА 11

DNS-сервер

Каждый компьютер, подключенный к сети, должен иметь свой адрес, чтобы его легко можно было найти и обмениваться с ним данными. Это как телефонные аппараты. Чтобы кому-нибудь позвонить, нужно знать его номер телефона, иначе коммутатор не сможет понять, с кем вас соединить.

Когда вы хотите получить Web-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. Вы отправляете письмо на определенный адрес, и если хотите, чтобы вам ответили, то должны указать обратный адрес.

В эпоху зарождения сетей компьютеров в ней было мало, поэтому для адресации был выбран самый простой вариант — числа. С увеличением количества компьютеров сетевая общественность стала осознавать, что помнить больше 20 адресов невозможно, поэтому было принято решение найти какой-либо способ, упрощающий запоминание.

Изменять принцип нумерации поздно, да и выбранная IP-система 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 уже по базе DNS находит нужный IP-адрес и по нему соединяется с компьютером/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 образом, Web-серверы фирм можно найти просто по их имени, и не надо запоминать лишнюю информацию.

11.1. Введение в DNS

В первое время преобразование IP-адресов в имена происходило с помощью простого текстового файла, по которому и проводилось сопоставление параметров. В Linux за это отвечает файл **/etc/hosts**. Несмотря на его слабости и неудобства в сопровождении, в малых сетях возможностей такого файла достаточно.

С расширением сети понадобился новый способ хранения соответствий. На смену файлам пришла доменная система имен (DNS, Domain Name System), предназначенная для преобразования символьного имени в IP-адрес и наоборот. Ее преимущества могут проявляться не только в Интернете, но и в локальных сетях с достаточно большим количеством компьютеров.

Внедрение DNS выявило еще одно преимущество этой системы — под одним и тем же IP-адресом может скрываться несколько узлов. Например, хостинговые компании на одном сервере могут содержать несколько сайтов.

Сервер DNS — это большая база данных, в которой хранятся сведения о соответствии имен узлов и IP-адресов. Самое главное, что эта информация децентрализована, и в Интернете существуют тысячи таких серверов.

База данных имеет иерархическую структуру, вершиной которой является точка ".". Это наподобие знака "/" в файловой системе Linux, обозначающего корневую директорию. На первом уровне идут домены типа COM, ORG, NET, GOV, RU, DE и т. д. Ниже находятся имена доменов второго уровня. Пример описанной структуры приведен на рис. 11.1.

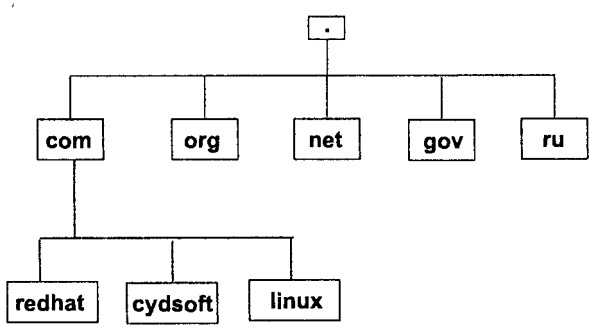

Рис. 11.1. Иерархия доменов

Допустим, что нужно найти IP-адрес сервера **www.cydsoft.com**. Имя разбирается справа налево. Сначала направляется запрос к корневому DNS-серверу, который должен указать, кто обслуживает домен **com**. Затем на найденном сервере по запросу осуществляется поиск домена с именем **cydsoft**. Если такой найден, то мы получим адрес DNS-сервера, который обслуживает

cydsoft.com. И уже ему направляется запрос на получение адреса домена **www.cydsoft.com**. Результатом будет IP-адрес сервера/компьютера, которому присвоено имя **www.cydsoft.com**.

Все эти действия происходят прозрачно для конечного пользователя, и когда вы набираете в браузере адрес, то никогда не увидите всех 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 достаточно много кэширующих серверов, в принципе, выполняющих свои функции автоматически. Достаточно только его установить, 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 может быть 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. Таким образом, 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 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. Результат: `77.15.1.190.in-addr.arpa`.

11.2. Локальный 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. Первая — это петля. Напоминаю, что во всех компьютерах имя `localhost` и IP-дрес 127.0.0.1 указывают на текущую машину. Это значит, что если нужно выполнить ping к локальному компьютеру, можно написать:

```
ping 127.0.0.1
```

Во второй записи устанавливается соответствие между заданным для вашего сетевого интерфейса IP-адреса и символьным именем. В данном случае моей сетевой карте присвоен адрес 192.168.77.1, а 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 данном 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 **http://www.redhat.com/**, при этом каждый раз происходит запрос к DNS-серверу, что может служить задержкой в пару секунд перед началом загрузки страницы. Чтобы ускорить этот процесс, можно вручную прописать в файл **/etc/hosts** следующую запись:

```
209.132.177.50    www.redhat.com
```

Внимание!

Адрес 209.132.177.50 действительно соответствует сайту **www.redhat.com** на момент написания 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 узнать 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, и 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 запроса к нему 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 стоит имя вашего компьютера. В этом параметре может быть перечислено несколько серверов, разделенных пробелами или символами табуляции. Например:

```
search FlenovM MyServer
```

Поиск на локальном сервере происходит достаточно 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 использовать 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 (на данный момент для большинства дистрибутивов не более 3). Они будут опрашиваться в порядке перечисления 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, и 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, не надо 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 завершатся.

На рис. 11.2 вы можете увидеть запущенное приложение для настройки DNS-сервиса. В центре показано окно, которое появляется по нажатию кнопки **До-**

бавить. Как видите, достаточно только выбрать тип зоны и ввести имя домена, и все готово.

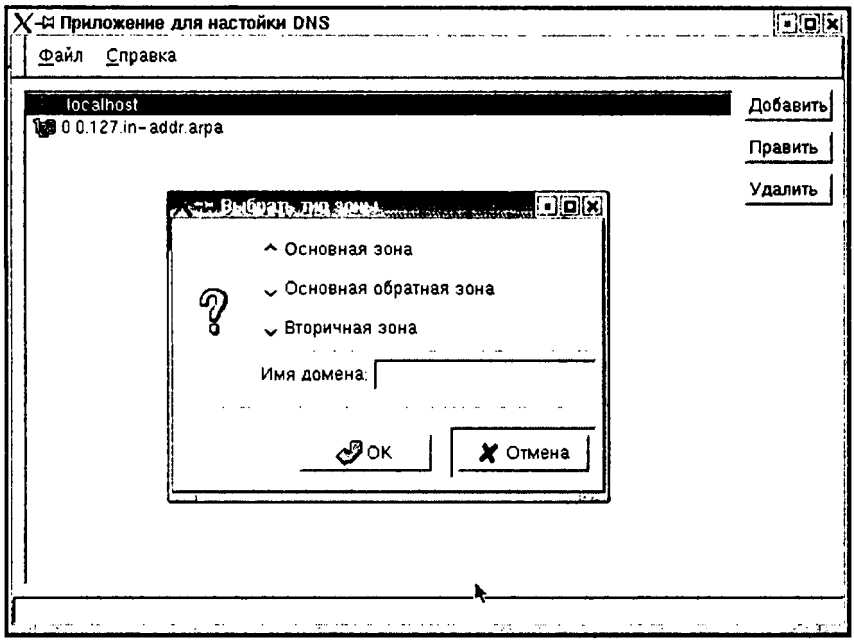

Рис. 11.2. Окно для настройки DNS-сервиса

Несмотря на наличие простой графической программы, мы рассмотрим работу DNS на примере конфигурационных файлов, которые могут использоваться сервисом. Их прямое редактирование позволит сделать более тонкую настройку, и вы лучше будете понимать процесс работы DNS.

Настройка DNS-сервиса начинается с файла **/etc/named.conf**. Пример его содержимого приведен в листинге 11.1.

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 данном примере файл разбит на четыре раздела, каждый из которых имеет следующий формат:

```
тип имя {
Параметр1;
Параметр2;
...
};
```

Давайте разберем назначение разделов.

Первым идет `options`:

```
options {
  directory "/var/named/";
};
```

В фигурных скобках только один параметр — `directory`, который указывает на домашнюю директорию DNS-сервера. Все его файлы будут располагаться там.

Остальные разделы имеют тип `zone` (и через пробел в кавычках стоит 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 описана следующим образом:

```
zone "." {
  type hint;
  file "named.ca";
};
```

Что это за зона в виде точки? Вспомните теорию DNS, которую мы рассматривали в начале главы. В базе данных DNS так обозначается корень. Получается, что раздел описывает корневую зону. Тип зоны `hint`, т. е. наш сервер, будет всего лишь хранить ссылки на DNS-серверы. Так как 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 может изменяться, и лучше всего получить последнюю версию с сервера **internic.net**. Для этого выполните команду:

```
dig @rs.internic.net . ns > named.ca
```

Перейдем к следующему разделу:

```
zone  "sitename.com" {
       type master;
       file  "sitename.zone";
};
```

Здесь описывается зона sitename.com. Тип записи `master`, значит ваш DNS-сервер будет главным, а все остальные будут только сверяться с ним и кэшировать информацию. Сведения об этой зоне будут храниться в файле sitename.zone рабочей директории. В нашем случае это **/var/named**.

Следующая зона описывает обратное преобразование IP-адресов 190.12.10.* в имена:

```
zone  "10.12.190.in-addr.arpa" {
       type master;
       file  "10.12.190.in-addr.arpa.zone";
};
```

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 директории **/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;
- sitename.zone — отвечает за преобразование имени sitename.com в IP-адрес;
- 10.12.190.in-addr.arpa.zone — отвечает за преобразование адресов сети 190.12.10.* в имена.

Файл sitename.zone может выглядеть следующим образом:

```
@    IN  SOA  ns.sitename.com root.sitename.com (
          1 ; serial
```

```
        28800 ; refresh
        7200 ; retry
        604800 ; expire
        86400 ; ttl
        )
IN    NS  ns.sitename.com.
IN    MX  10 mail.sitename.com.
ns    A  190.12.10.1
mail   A  190.12.10.2
```

Разберем основные типы записей, которые используются при конфигурировании DNS:

- SOA (Start of Authority, Начало полномочий) — основная информация, которая включает в себя почтовый адрес администратора, а также время жизни записи в кэше, данные о частоте ее обновления и др.;
- A (Address, Адрес) — доменное имя и IP-адрес компьютера;
- CNAME (Canonical Name, Каноническое имя) — синоним для реального доменного имени, которое указано в записи типа A;
- PTR (Pointer, Указатель) — отображение доменного имени по его IP-адресу;
- TXT (Text, Текст) — дополнительная информация, которая может содержать любое описание;
- RP (Responsible Person, Ответственное лицо) — E-mail-адрес ответственного за работу человека;
- HINFO (Host Information) — информация о компьютере, такая как описание ОС и установленного оборудования.

В целях безопасности записи HINFO и TXT не используют. Ничего лишнего хакеру не должно быть доступно, тем более не стоит вводить информацию о компьютере и его ОС. Записи HINFO и TXT чисто информационные и не несут в себе никаких полезных данных, способных повлиять на работу сервера.

Теперь вернемся к файлу sitename.zone и рассмотрим его содержимое. В первой строке (тип SOA) идет описание зоны. Сначала указывается имя DNS-сервера (`ns.sitename.com`) и человека, ответственного за запись (`root@sitename.com`). В скобках перечислены параметры, которые для удобства расположены каждый в своей строке. Первым идет номер записи. После каждой корректировки увеличивайте это значение на 1 или записывайте туда дату последнего редактирования. По этому значению другие серверы будут узнавать, было ли изменение записи.

Следующий параметр `refresh` — частота, с которой другие серверы должны обновлять свою информацию. В случае ошибки сервер должен повторить попытку через время, указанное в третьем параметре (`retry`).

Последний параметр (`ttl`) устанавливает минимальное время жизни записи на кэширующих серверах, т. е. определяет, когда информация о зоне на кэширующем сервере будет считаться недействительной.

По этим параметрам остальные DNS-серверы будут знать, как себя вести для обновления информации о зоне, которую контролирует ваш DNS-сервер.

Следующая строка имеет тип `NS`, и таких записей может быть несколько. Сокращение NS в данном случае означает Name Server. Здесь описываются DNS-серверы, которые отвечают за эту зону. Именно через эти серверы все остальные участники будут преобразовывать символьное имя **sitename.com** в IP-адрес.

После этого могут идти записи `MX` (Mail eXchange). По ним серверы определяют, куда отправлять почту, которая приходит на домен **sitename.com**. В нашем примере это сервер **mail.sitename.com**, а число перед его именем — это приоритет. Если в файле будет несколько записей `MX`, то они будут использоваться в соответствии с приоритетом.

Внимание!

В записях типа `NS` и `MX` в конце адреса обязательно должна быть точка!

И наконец, строки преобразования. Они выглядят следующим образом:

```
имя    A    адрес
```

В нашем примере две строки:

```
ns     A    190.12.10.1
mail   A    190.12.10.2
```

Это значит, что имена **ns.servername.com** и **mail.servername.com** соответствуют IP-адресу 190.12.10.1.

11.6. Обратная зона

Теперь рассмотрим файл описания обратного преобразования IP-адреса в имя (10.12.190.in-addr.arpa.zone). Он может иметь примерно следующий вид:

```
@    IN  SOA  ns.sitename.com root.sitename.com (
         1 ; serial
         28800 ; refresh
         7200 ; retry
```

```
        604800 ; expire
        86400 ; ttk
        )

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 числом, стоящим 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 ничего сверх страшного в ней нет, и хакер не сможет 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-адрес, поэтому падение DNS для них смертельно.

Помимо вывода из строя сервера DNS может предоставлять хакеру слишком много информации, из которой он сможет узнать структуру сети. Чтобы этого не произошло, желательно использовать два DNS-сервера:

1. Общедоступный, содержащий необходимые строки для работы удаленных пользователей с общими ресурсами.
2. Локальный, видимый только пользователям вашей сети и содержащий все необходимые записи для их работы.

На локальном сервере можно так настроить сетевой экран, чтобы он воспринимал только внутренний 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 могут спать спокойным сном, пока их охраняет сетевой экран.

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 очень требовательны к оборудованию. В моей сети работает 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 такой древней техники более чем достаточно и хватит на ближайшие годы. Таким образом, старому компьютеру можно дать новую жизнь, и довольно долгую, а главное, что для компании такое решение окажется приемлемым по цене.

Но технология создания вторичных серверов опасна. С помощью команды `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 директиву:

```
host -l server.com ns1.server.com
```

В ответ на это 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 описании 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 DNS-записи.

В остальном защита DNS и любых других сервисов и ОС Linux идентичны. 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, поэтому под эту процедуру попадают все серверы, в том числе и DNS.

ГЛАВА 12

Мониторинг системы

Первоначальная задача администратора — установить систему, правильно распределить права доступа и настроить все необходимые сервисы. После этого многие из них складывают ручки и начинают гонять монстров по коридорам виртуального мира Doom 3. Если вы являетесь таким администратором, то рано или поздно ваша система будет взломана, и последствия будут плачевными.

Чтобы уменьшить вероятность проникновения в систему постороннего или обезопасить себя от недобросовестных пользователей вашей сети, нужно постоянно держать сервер под контролем. Большинство взломов происходит благодаря тому, что администраторы не успевают обновить какие-либо сервисы и установить заплатки. Очень часто взломщики узнают об уязвимости раньше и начинают ломать все серверы с этой ошибкой, которые попадаются на глаза.

Хороший администратор может и должен найти информацию об уязвимости и сделать все необходимое для предотвращения любой атаки. Для этого должен производиться регулярный мониторинг системы и поиск узких мест. Иногда хакеры проникают в систему и долгое время находятся в ней, не проявляя видимой активности. Вы должны уметь найти таких мышек и обезвредить до того, как они смогут натворить что-либо печальное.

Если взлом произошел, то ваша задача не просто восстановить работу, а предотвратить повторное вмешательство. Я много раз видел людей, которые после взлома просто восстанавливают удаленные файлы и продолжают заниматься своими делами в надежде, что такое не повторится. Это ошибка, потому что хакер уже почувствовал себя безнаказанным и обязательно вернется.

К следующему визиту вы должны быть готовы. Сделайте все возможное, чтобы найти сведения о взломщике, о том, как он проникает на сервер, и постарайтесь блокировать атаку. Вы также должны просмотреть все последние

бюллетени ошибок (BugTraq) в поисках информации о дырах в вашей версии ОС и установленных сервисах.

Не будем дожидаться, когда злоумышленник проникнет в систему, и мы потеряем сон. Давайте рассмотрим действия, которые можно произвести еще до взлома, когда система работает нормально. Это поможет повысить безопасность сервера.

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, где его не должно быть), и вы должны регулярно сканировать систему на предмет безопасности описанными ниже методами. Особенно это касается новой системы (сразу после инсталляции и настройки), обновления, добавления программ или сразу после взлома.

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 посоветовал посещать **www.securityfocus.com**, чтобы следить за новостями, и сейчас не отказываюсь от своих слов. Новинки действительно надо смотреть на подобных серверах, но ведь есть еще ворох старых уязвимостей, которые могли быть и не залатаны на сервере. Как же поступить с ними? Неужели придется качать все сплоиты и руками проверять каждую дыру? Ну, конечно же, нет. Существует громадное количество программ для автоматизации тестирования сервера на ошибки, и самые распространенные — SATAN, Internet Scanner, NetSonar, CyberCop Scanner.

Я не стану рекомендовать какую-нибудь определенную программу. Не существует такой утилиты, в которой была бы база абсолютно всех потенциальных уязвимостей. Поэтому скачивайте все, что попадется под руку, и тестируйте систему всеми доступными программами. Возможно, что-то вам и пригодится. Но обязательно обратите внимание на продукты компании Internet Security Systems (ISS, системы Интернет безопасности **www.iss.net**), потому что сканеры этой фирмы (Internet Scanner, Security Manager, System Scanner и Database Scanner) используют все три метода сканирования, о чем мы поговорим чуть позже. Сотрудники ISS работают в тесном контакте с Microsoft и постоянно обновляют базу данных уязвимостей. Но несмотря на то, что продукты этой фирмы лучшие, я советую использовать хотя бы еще один сканер другого производителя.

Компания Internet Security Systems разработала целый комплект утилит под общим названием SAFEsuite. В него входят не только компоненты проверки безопасности системы, но и модули выявления вторжения и оценки конфигурации основных серверных ОС.

Сканеры безопасности как антивирусы — защищают хорошо, но только от старых приемов. Любой новый метод взлома не будет обнаружен, пока вы не обновите программу. Поэтому я не рекомендую целиком и полностью полагаться на отчеты автоматизированного сканирования, а после работы программы самостоятельно проверить наличие последних уязвимостей, описанных в каком-либо бюллетене ошибок (BugTraq).

С помощью автоматизированного контроля очень хорошо производить первоначальную проверку, чтобы убедиться в отсутствии старых ляпсусов. Если ошибки найдены, то нужно обновить уязвимую программу/ОС/Сервис или поискать на том же сайте **www.securityfocus.com** способ обезвреживания. Почти всегда вместе с описанием уязвимости дается вакцина, позволяющая залатать прореху в сервисе или ОС. Вакцину может предложить и программа сканирования, если в базе данных есть решение проблемы для данного случая.

Почему даже после лучшего и самого полного сканирования нельзя быть уверенным, что уязвимостей нет? Помимо новых ошибок в сервере надо принять во внимание еще и фактор конфигурации. На каждом сервере могут быть свои настройки, и при определенных условиях легко находимая вручную уязвимость может остаться незаметной для автоматического сканирования. На сканер надейся, а сам — не плошай. Так что, продолжайте тестировать систему на известные вам ошибки самостоятельно.

Каждый сканер использует свои способы и приемы, и если один из них не нашел ошибок, то другой может отыскать. Специалисты по безопасности любят приводить пример с квартирой. Допустим, что вы пришли к другу и позвонили в дверь, но никто не открыл. Это не значит, что дома никого нет, просто хозяин мог не услышать звонок, или он не работал. Но если позвонить по телефону, который лежит в этот момент возле хозяина, то он возьмет трубку. Может быть и обратная ситуация, когда вы названиваете по телефону, но его не слышно, а на звонок в дверь домочадцы отреагируют.

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.

Существует три метода автоматического определения уязвимости: сканирование, зондирование и имитация. В первом случае сканер собирает информацию о сервере, проверяет порты, чтобы узнать, какие установлены серви-

сы/демоны, и на основе их выдает отчет о потенциальных ошибках. Например, сканер может проверить сервер и увидеть на 21 порту 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 есть уязвимость для данного сервера, то пользователю выдается соответствующее сообщение.

Сканирование далеко не самый точный процесс, потому что автоматическое определение легко обмануть, да и уязвимости может не быть. Некоторые погрешности в сервисах проявляются только при определенных настройках, т. е. при установленных вами параметрах ошибка не обнаружится.

При зондировании сканер не обследует порты, а проверяет программы на наличие в них уязвимого кода. Этот процесс похож на работу антивируса, который просматривает все программы на наличие соответствующего кода. Ситуации похожие, но искомые объекты разные.

Метод хорош, но одна и та же ошибка может встречаться в нескольких программах. И если код в них разный, то сканер ее не обнаружит.

Во время 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 попробует выполнить инструкцию. Конечно же, это приведет к зависанию, но вы точно будете знать о наличии или отсутствии ошибки на нем.

Имитация — самый долгий, но надежный способ, потому что если программе удалось взломать какой-либо сервис, то и у хакера это получится. При установке нового FTP-сервера, который еще незнаком сканерам, он будет опробован на уже известные ошибки других серверов. Очень часто программисты разных фирм допускают одни и те же промахи, при этом методом сканирования анализатор может не найти подобную уязвимость только потому, что для данной версии нет записей в базе данных.

Когда 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 Firewall, то уязвимость станет опасной.

Дайте сканеру все необходимые права на доступ к тестируемой системе. Например, некоторые считают, что наиболее эффективно удаленное сканирование, когда по сети имитируется атака. Это правильно, но сколько времени понадобится на проверку стойкости паролей для учетных записей? Очень много! А сканирование реестра и файловой системы станет невозможным. Поэтому локальный контроль может дать более качественный результат.

При дистанционном сканировании только производится попытка прорваться в сеть. Такой анализ может указать на стойкость защиты от нападения извне. Но по статистике большинство взломов происходит изнутри, когда зарегистрированный пользователь поднимает свои права и тем самым получает доступ к запрещенной информации. Хакер тоже может иметь какую-нибудь учетную запись с минимальным статусом и воспользоваться уязвимостями для повышения прав доступа. Поэтому сканирование должно происходить и дистанционно для обнаружения потайных дверей и локально для выявления ошибок в конфигурации, с помощью которых можно изменить привилегии.

Автоматические сканеры проверяют не только уязвимости ОС и ее сервисов, но и сложность пароля, и имена учетных записей. В анализаторах есть база наиболее часто используемых имен и паролей, и программа перебором проверяет их. Если удалось проникнуть в систему, то выдается сообщение о слишком простом пароле. Обязательно замените его, потому что хакер может использовать тот же метод, и легко узнает параметры учетной записи.

Анализаторы безопасности могут использовать как хакеры, так и администраторы. Но задачи у них разные. Одним нужно автоматическое выявление ошибок для последующего применения, а вторые используют метод с целью закрытия уязвимости, причем желательно это сделать раньше, чем найдет и будет использовать дыры хакер.

В дальнейшем мы рассмотрим методы ручной проверки безопасности системы и программы, упрощающие этот процесс. Что использовать? Я же сказал, что все. Вы должны проверять систему всеми возможными методами. Ограничившись только одним способом, рискуете что-то упустить и оставить потенциальную лазейку.

12.2. Закрываем SUID- и SGID-двери

Как администратор или специалист по безопасности, вы должны знать свою систему от и до. Мы уже говорили, что одной из проблем может стать бит SUID или SGID. Вы должны вычистить все эти биты у программ, которыми не пользуетесь. Но как их найти? Для этого можно воспользоваться командой:

```
find / \( -perm -02000 -o -perm -04000 \) -ls
```

Эта директива найдет все файлы, у которых установлены права 02000 или 04000, что соответствует битам SUID и SGID. Выполнив команду, вы увидите на экране примерно следующий список:

```
130337 64 -rwsr-xr-x  1 root root  60104 Jul 29 2002 /bin/mount
130338 32 -rwsr-xr-x  1 root root  30664 Jul 29 2002 /bin/umount
```

```
130341 36 -rwsr-xr-x  1 root root  35040 Jul 19 2002 /bin/ping
130365 20 -rwsr-xr-x  1 root root  19072 Jul 10 2002 /bin/su
...
```

Самое страшное, что все программы из этого списка принадлежат пользователю root и будут выполняться от его имени. В системе есть файлы с SUID- и SGID-битом, выполняющиеся от имени других пользователей, но таких меньшинство.

Если вы видите, что программа не используется вами, то ее стоит удалить или снять биты. Если таких программ по вашему мнению нет, то подумайте еще раз. Может быть стоит от чего-то отказаться? Например, программа ping не является обязательной для сервера, и у нее бит SUID можно снять.

Если после корректировки привилегированных программ осталось много, то советую для начала убрать бит со всех программ. Конечно же, тогда пользователи не смогут монтировать устройства или менять себе пароль. А это им надо? Если уж возникнет острая необходимость, то всегда можно вернуть им такую возможность, восстановив SUID-бит.

А ведь можно сделать владельцем программы другую учетную запись, у которой будет меньше прав. Это сложно в реализации, потому что придется вручную изменять разрешения, но это сделает ваш сон более спокойным.

Почему необходимо регулярно проверять файлы на наличие SUID- или SGID-битов? Взломщики, проникая в систему, очень часто стремятся укрепиться в ней, сесть незаметно 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.

12.3. Проверка конфигурации

Мы рассмотрели достаточно много правил конфигурирования системы, и помнить все невозможно. Настройка системы достаточно сложный процесс, во время которого очень легко ошибиться. Но т. к. есть определенные правила конфигурирования, т. е. и возможность автоматизировать проверку.

Как мы уже знаем, есть программы, которые могут проверять конфигурацию. Это должно происходить на компьютере локально. В настоящее время уже написано достаточно много утилит для автоматизации контроля. Некоторые из них устарели и давно не обновлялись, а какие-то появились недавно и только еще наращивают свою функциональность (количество проверок).

12.3.1. lsat

Первая программа, с которой нам предстоит познакомиться, — lsat. Ее история не слишком длинная, но за счет частого обновления возможности программы быстро выросли, а благодаря модульной архитектуре расширение функций происходит легко и быстро.

Программа lsat поставляется в исходных кодах, и найти ее можно по адресу **http://usat.sourceforge.net/**. На момент написания этой книги была доступна версия 0.9.2. Скачайте архив в свою домашнюю директорию, и давайте рассмотрим, как установить и использовать программу. На сайте доступны tgz- и zip-версия. Я рекомендую воспользоваться первой, потому что это родной архив для Linux и проще в использовании.

Для установки выполните следующие команды:

```
tar xzvf lsat-0.9.2.tgz
./lsat-0.9.2.tgz/configure
./lsat-0.9.2.tgz/make
```

Первая директива распаковывает архив lsat-0.9.2.tgz. Имя файла может отличаться, если у вас другая версия программы. Вторая команда запускает конфигурирование, а третья — собирает проект из исходных кодов в один исполняемый файл.

Для запуска программы используйте следующую команду:

```
./lsat-0.9.2.tgz/lsat
```

Теперь можно идти готовить кофе и медленно и печально пить его. Процесс тестирования системы занимает довольно много времени, особенно на слабых машинах. При запуске можно указать один из следующих ключей:

- `-o имя файла` — отчет записывать в указанный файл. По умолчанию отчет попадает в lsat.out;
- `-v` — выдавать подробный отчет;
- `-s` — не выдавать никаких сообщений на экран. Это удобно при выполнении команды через cron;
- `-r` — выполнять проверку контрольных сумм (это делается с помощью RPM). Это позволяет выявить незаконно измененные программы.

Лучше всего lsat будет работать на Red Hat-подобных системах, потому что в нее встроена возможность работы с RPM-пакетами, которые являются отличительной особенностью дистрибутива Red Hat Linux и его клонов.

Во время проверки вы увидите примерно следующий текст на экране:

```
Starting LSAT...
Getting system information...
Running modules...
Running checkpkgs module...
...
...
Running checkx module...
Running checkftp module...
Finished.
Check lsat.out for details.
Don't forget to check your umask or file perms
when modifying files on the system.
```

Пока слова о безопасности отсутствуют. Только информация о том, что сканировалось. Все самое интересное после тестирования можно найти в файле **./lsat-0.9.2.tgz/lsat.out**. Я прогнал эту программу на системе сразу после установки и получил документ размером 190 Кбайт. Таким образом, есть над чем подумать и что изучать в вашей системе.

В 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 предлагает его подчистить.

В выходном файле мне очень понравилась надпись:

```
default init level is not set to 5. Good.
(уровень загрузки по умолчанию не равен 5. Хорошо.)
```

Мой уровень загрузки равен 5 (текстовый режим) и программа сообщает, что это хорошо. Разработчик lstat посчитал, что загрузка в графическом режиме

хуже для безопасности. Действительно, это лишние программы и дополнительные проблемы. Текстовый режим не требует столько ресурсов, работает меньше утилит, а значит, он быстрее и безопаснее.

Если немного опуститься ниже, то вы увидите список всех файлов в системе с установленными битами SUID и SGID. 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, которые имеют право изменять любые пользователи системы, даже с минимальными правами.

Далее идет список файлов, в которые могут писать пользователи каких-либо групп. Проверьте, возможно, не всем пользователям нужно давать такой статус. В идеале таких записей вообще не должно быть. В любой файл должны иметь право писать только владельцы или, в крайнем случае, пользователи группы, но никак не все.

Отчеты удобны и легко читаются, но в самом конце появляется ложка дегтя. Пусть она и небольшая, но она есть. Программа показывает изменения в файловой системе с момента последнего запуска. Вот тут разобраться с чем-либо очень сложно. Информация выводится как угодно, а ведь удобнее было разделить модификации по степени опасности. Например, удаленный или добавленный в разделе **/tmp** файл не так важен, потому что там изменения происходят тоннами каждые пять минут. А вот все, что касается раздела **/etc**, намного опаснее, и эти записи нужно выделять.

12.3.2. bastille

Проект bastille (**http://bastille-linux.sourceforge.net/**) существует уже давно и создан специалистами по безопасности Linux. Разработчики собирались написать свою версию ОС, которая будет более безопасной, но, видимо, не рассчитали свои силы. Глядя на bastille, так и хочется сказать: "Жаль!!!".

Программа проверяет систему и выдает отчет, из которого вы можете узнать о найденных слабостях в ОС, и если пожелаете, то bastille автоматически примет необходимые меры по устранению уязвимости.

Работа программы настолько проста и удобна, что я даже не буду ее описывать. В отличие от подобных средств, bastille может работать не только в текстовом, но и в графическом режиме. Для установки программы можно воспользоваться RPM-архивом или скомпилировать файлы из исходного кода.

12.4. Выявление атак

Хороший администратор должен сделать все, чтобы убить попытку атаки на свою систему еще в зародыше. Давайте вспомним, с чего начинается взлом системы? Конечно, со сбора информации об интересующем компьютере или сервере, и это мы рассмотрели в самом начале книги. Хакер пытается узнать о системе все, что только можно, а администратор должен сделать так, чтобы предотвратить этот процесс или запутать взломщика.

Самый простой и один из наиболее информативных методов — сканирование портов. Чтобы выяснить, кто, когда и откуда произвел попытку проникновения, необходимо отлавливать любые нестандартные события портов. Конечно же, вручную это сделать сложно, поэтому лучше запастись хорошей программой.

Средства автоматического выявления атак достаточно хороши, но не всегда приемлемы. Например, если ваш сервер популярен, то количество сканирований в день будет довольно большим. Я думаю, что такие узлы, как **www.yahoo.com** или **www.microsoft.com**, сканируют тысячи, а то и миллионы раз в день, и на каждую попытку обращать внимание бесполезно. А самое главное, автоматическое выявление атак требует ресурсы, а иногда и немалые. Если попытаться протоколировать каждое сканирование, то злоумышленник может сделать такие пакеты, которые будут имитировать атаки, и тогда вся вычислительная мощь сервера уйдет на выявление этих наскоков. Получится классический отказ от обслуживания (DoS), потому что сервер уже не будет обрабатывать запросы клиентов.

Но если у вас локальный сервер в организации или просто домашний компьютер в небольшой сети, то определение любителей сканирования помогут заведомо предотвратить взлом.

Еще один недостаток автоматического определения атак — вы сами не сможете воспользоваться утилитами сканирования безопасности своих серверов, потому что это будет воспринято как нападение. Когда вы сами сканируете систему, то обязательно отключайте программы обнаружения атак, иначе ваши действия не принесут результата.

12.4.1. Klaxon

Самой простой и эффективной является утилита Klaxon (**http://www.eng.auburn.edu/users/doug/second.html**). Она следит за неиспользуемыми системой портами и при обращении на них пытается определить всю возможную информацию о сканирующем IP-адресе и сохранить ее в журнале.

Программа устанавливается достаточно просто. В конфигурационный файл **/etc/inetd.conf** нужно добавить строки, как показано в примере:

```
#
# Local testing counterintelligence
#
rexec  stream tcp nowait root  /etc/local/klaxon klaxon rexec
link  stream tcp nowait root  /etc/local/klaxon klaxon link
supdup stream tcp nowait root  /etc/local/klaxon klaxon supdup
tftp  dgram  udp wait  root  /etc/local/klaxon klaxon tftp
```

Подразумевается, что программа установлена в директории **/etc/local/klaxon**. Описанные в конфигурационном файле сервисы перенаправляются на klaxon, и вы сможете контролировать, когда и кто пытался обратиться к ним. Реально эти сервисы не должны использоваться в системе, и соответствующий порт открывает программа klaxon, и если обращение все же было, то это, скорее всего, сканирование или даже попытка взлома. Просто так на неиспользуемый порт никто подключаться не будет.

Такой метод хорош тем, что, например, сервис rexec (удаленное выполнение команд) не нужен пользователям, и его очень часто ищут хакеры для проникновения в систему. Если с определенного адреса была попытка (пусть и неудачная) обратиться к rexec, можно взять на заметку это событие и этот IP и уделять ему больше внимания.

Я рекомендую установить klaxon в системе не более чем на 2—3 сервиса, потому что слишком большое количество портов вызовет подозрение у взломщика. К тому же, если klaxon возьмет на обслуживание более 5 портов, то многократное сканирование может отнять лишние ресурсы системы. Это создаст предпосылки для проведения успешной атаки отказа от обслуживания.

12.4.2. PortSentry

Раньше эта программа принадлежала компании Psionic, но сейчас ссылка **www.psionic.com** ведет на один из разделов компании Cisco, и утилиты там уже нет. Но в Интернете она еще осталась, и полный исходный код можно взять с сайта **http://sourceforge.net/projects/sentrytools**.

Так как программа поставляется в исходных кодах, то для ее установки требуется распаковка архива и компиляция. Это уже не должно у вас вызывать проблем.

Для разархивирования выполняем команду:

```
tar xzvf portsentry-1.2.tar.gz
```

В моем случае создалась директория **portsentry_beta**. У вас это имя может быть другим из-за несовпадения версии программы, возможно, изменившейся к моменту выхода книги. Имя каталога будет видно в процессе разархивирования, т. к. программа отображает на экране список файлов в виде `каталог/имя файла`.

Перейдите в созданный каталог с исходными кодами, чтобы выполнять команды в нем:

```
cd portsentry_beta
```

Теперь поговорим о компиляции. Программа PortSentry написана универсально и может работать в разных Unix-подобных системах, и помимо Linux это может быть Solaris, FreeBSD, OpenBSD и т. д. При компиляции вы должны явно указать, какая ОС у вас установлена:

```
make linux
```

Теперь собственно установка. По умолчанию файлы программы копируются в директорию **/usr/local/psionic**, но этим можно управлять, если в файле makefile поменять параметр `INSTALLDIR`. Если все устраивает, то выполняем команду:

```
make install
```

Затем нужно установить описанным для PortSentry способом программу Logcheck, поэтому приведу только команды:

```
tar xzvf logcheck-1.1.1.tar.gz
cd logcheck-1.1.1
make linux
make install
```

Эта программа по умолчанию устанавливается в директорию **/usr/local/etc**. Каталог также можно изменить, отредактировав параметр `INSTALLDIR` в файле makefile.

Все настройки программы PortSentry находятся в файле **/usr/local/psionic /portsentry/prtsentry.conf**. По умолчанию все закомментировано, и вам необходимо только открыть нужные строки.

Например, вы хотите, чтобы программа следила за определенными портами. На этот случай в конфигурационном файле подготовлены закомментирован-

ные записи для разных типов серверов. Выберите нижний и уберите в начале строки знак "#". Тем самым вы укажете порты для наблюдения:

```
TCP_PORTS="1,11,15,79,111,119,143,540,635,1080,1524,2000,5742,6667"
UDP_PORTS="1,7,9,69,161,162,513,635,640,641,700,32770,32771,32772"
```

Помимо мониторинга в программе есть отличная способность — при выявлении попытки атаки конфигурировать Firewall на запрет любого трафика с компьютером, который пытался взломать систему. Но это тоже по умолчанию отключено, и чтобы воспользоваться этой возможностью, нужно убрать комментарий со строки, соответствующей вашему серверу.

Мы рассматриваем Linux, а в нем, чаще всего, используют Firewall ipchains. Для него нужна запись:

```
KILL_ROUTE="/sbin/ipchains -I input -s $TARGET$ -j DENY -l"
```

Убедитесь только, что программа сетевого экрана установлена по указанному пути (**/sbin/ipchains**). Для этого можно выполнить команду поиска программы:

```
which ipchains
```

Если в вашей системе применяется iptables, то нужно использовать строку:

```
KILL_ROUTE="/usr/local/bin/iptables -I INPUT -s $TARGET$ -j DROP"
```

Я считаю возможность выявления атаки и автоматического конфигурирования сетевого экрана очень мощной. В то же время, любая программа может ошибиться и запретить доступ не тому, кому надо. Например, взломщик может имитировать атаку от имени другого пользователя (скажем, босса), и тогда PortSentry запретит шефу доступ к ресурсам сервера. А это уже не есть хорошо.

Я попытался в своей тестовой системе закидать сервер пакетами подключения к различным портам. При этом в качестве IP-отправителя подставлял чужие адреса, в результате чего сервер стал недоступным для них. Вы должны контролировать все, что прописывает в сетевом экране программа мониторинга, иначе хакер может забросать систему запросами так, что она запретит доступ для всех компьютеров вашей сети.

Для запуска программы мониторинга выполните команды:

```
/usr/local/psionic/portsentry/portsentry -atcp
/usr/local/psionic/portsentry/portsentry -audp
```

Первая команда запускает мониторинг TCP-портов, а вторая — заставит программу наблюдать за UDP-портами. Вся активность будет сохраняться в журнале, который можно проверить с помощью установленной нами ранее программы Logcheck. Я рекомендую поместить эту программу в задания,

чтобы она выполнялась через определенные интервалы времени (не менее 15 минут) и сообщала администратору о событиях в системе.

Для начала желательно сконфигурировать программу Logcheck. Для этого откройте файл **/usr/local/etc/logcheck.sh** и добавьте в него следующую строку (если ее нет):

```
"mailto:SYSADMIN=admin @server.com"
```

Здесь `admin@server.com` — это ваш E-mail-адрес, на который нужно отправлять электронные сообщения с информацией о сохраненных программой PortSentry в журнале записях. Теперь остается только сделать так, чтобы сценарий **/usr/local/etc/logcheck.sh** выполнялся через определенные промежутки времени. Для этого подойдет программа crontab.

Для тестирования программы PortSentry я сконфигурировал ее, как показано выше, и запустил прослушивание портов. Сканер CyD NET Utils (**http://www.cydsoft.com**) показал только первые два открытых канала. Все остальные оказались для программы закрыты, хотя реально их больше двух. Я сел за клавиатуру своего Linux-сервера и выполнил команду `cat /etc/hosts.deny`, чтобы увидеть файл **/etc/hosts.deny**, который содержит IP-адреса всех компьютеров, которым запрещено подключаться к серверу.

На экране появилось содержимое этого файла, и последней строкой был IP-адрес компьютера, с которого я производил сканирование:

```
ALL:    192.168.77.10
```

Программа PortSentry среагировала достаточно быстро и эффективно, прописав в файл **/etc/hosts.deny** запрет использования любого сервиса с адреса 192.168.77.10. Больше я уже ничего сделать не мог. Единственный способ снова получить доступ — удаление строки из файла **/etc/hosts.deny**, показанной выше.

Нужно заметить, что некоторые порты могут использоваться достаточно часто, и программа будет думать, что это попытки взлома. К ним относятся ident (113) и NetBIOS-порты (39), и их лучше всего исключить из наблюдения. Для этого в конфигурационном файле **/usr/local/psionic/portsentry/prtsentry.conf** найдите строки `ADVANCED_EXCLUDE_TCP` (для TCP-портов) и `ADVANCED_EXCLUDE_UDP` (для UDP-портов) и добавьте нужные каналы в список. По умолчанию в программе исключены следующие порты:

```
ADVANCED_EXCLUDE_TCP="113,139"
ADVANCED_EXCLUDE_UDP="520,138,137,67"
```

Как видите, 113 и 139 порты по умолчанию уже исключаются из мониторинга.

12.4.3. LIDS

Пакет LIDS (Linux Intrusion Detection/Defence System, определение вторжения в Linux и система безопасности). Несмотря на то, что я не очень люблю использовать патчи ядра, этот заслуживает вашего внимания, потому что обладает большими возможностями и реально позволяет повысить безопасность системы.

Конфигурация и работа системы надежно защищены. Так, например, файлы настройки зашифрованы, и внести в них изменения достаточно сложно. Остановить работу LIDS также проблематично, потому что для этого необходимо знать пароль администратора системы.

Функция определения попыток сканирования — это самая малость, что может делать этот пакет. Кроме того, LIDS позволяет ограничить доступ к файлам на уровне программ, а не пользователей, тем самым расширяются возможности по управлению правами доступа и увеличивается общая безопасность. Например, командам `ls`, `cat` и текстовым редакторам можно категорически запретить работу с каталогом **/etc**. Таким образом, мы усложняем хакеру задачу по просмотру файла паролей **/etc/passwd**.

Установка LIDS не из простых, потому что требуется инсталляция патча на исходные коды и перекомпиляция ядра Linux. И вот тут на первый план выходит неудобство патча — при его обновлении нет гарантии, что он будет работать верно или не нарушит функционирование ядра. Исходные коды могут быть испорчены, и компиляция станет невозможной. При выходе новой версии ядра приходится на тестовой машине проверять его работу и устанавливать все на свой страх и риск. А если не обновлять ядро, то нет уверенности, что в дальнейшем оно будет удовлетворять новым требованиям безопасности.

Более подробную информацию по работе пакета LIDS вы можете узнать на официальном сайте **www.lids.org**.

12.5. Журналирование

В Linux работает одновременно сразу несколько журналов, и по содержащейся в них информации можно узнать много интересного. По ним вы сможете вычислить хакера и увидеть, когда он проник в систему, и немало всего любопытного. Так как ведение журналов — это один из инструментов обеспечения безопасности, мы рассмотрим данный вопрос достаточно подробно. Это позволит вам лучше контролировать свои владения.

12.5.1. Основные команды

Информация о текущих пользователях системы сохраняется в файле **/var/run/utmp**. Если попытаться посмотреть его содержимое, например, командой `cat`, то ничего хорошего нашему взору не откроется. Этот журнал хранит данные не в текстовом, а бинарном виде, и получение информации возможно только с помощью специализированных программ (команд).

who

Команда `who` позволяет узнать, кто сейчас зарегистрирован в системе и сколько времени в ней находится. Информация достается из файла **/var/run/utmp**. Введите эту команду, и на экране появится список примерно следующего вида:

```
robert    tty1    Dec 8 10:15
root      tty2    Dec 8 11:07
```

Из этого списка становится ясно, что пользователь robert работает за первым терминалом (`tty1`) и вошел в систему 8 декабря в 10 часов 15 минут.

Большинство хакеров при входе в систему выполняют эту команду, чтобы выяснить, есть ли сейчас в системе администратор. Если пользователь root присутствует, то начинающие хакеры стараются уйти, т. к. опасаются, что их знаний не хватит, чтобы остаться незамеченными.

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, равный нулю. Она позволяет получить доступ ко всей системе, и при этом имеет имя отличное от root, и не вызовет подозрений, когда я буду работать. Так что в моем случае никогда нельзя увидеть пользователя root.

users

Эта команда позволяет вытащить из журнала **/var/run/utmp** список всех пользователей, которые сейчас зарегистрированы в системе.

В журнале **/var/run/utmp** информация хранится временно, только на момент присутствия пользователя. Когда он выходит из системы, соответствующая запись удаляется. После этого выяснить, кто и когда работал, можно только по журналу **/var/log/wtmp**. Это также бинарный файл, поэтому его содержимое можно увидеть с помощью специализированных программ.

last

Команда позволяет выяснить, когда и сколько времени определенный пользователь находился в системе. В качестве параметра 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 список:

```
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 входили в систему.

Если выполнить эту команду для себя, то может вывалиться такой список, что читать его будет невозможно, потому что вы достаточно 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
```

lastlog

Если выполнить команду `lastlog`, то она выведет на экран перечень всех пользователей с датами их последнего подключения к системе. Пример списка можно увидеть в листинге 12.1.

Листинг 12.1. Результат выполнения команды `lastlog`

```
Username         Port     From             Latest
root             ftpd2022 192.168.77.10    Mon Feb 21 12:05:06 +0300 2005
bin                                        **Never logged in**
daemon                                     **Never logged in**
adm                                        **Never logged in**
lp                                         **Never logged in**
sync                                       **Never logged in**
shutdown                                   **Never logged in**
halt                                       **Never logged in**
mail                                       **Never logged in**
news                                       **Never logged in**
uucp                                       **Never logged in**
```

```
operator                                **Never logged in**
games                                   **Never logged in**
gopher                                  **Never logged in**
ftp                                     **Never logged in**
nobody                                  **Never logged in**
vcsa                                    **Never logged in**
mailnull                                **Never logged in**
rpm                                     **Never logged in**
xfs                                     **Never logged in**
apache                                  **Never logged in**
ntp                                     **Never logged in**
rpc                                     **Never logged in**
gdm                                     **Never logged in**
rpcuser                                 **Never logged in**
nscd                                    **Never logged in**
ident                                   **Never logged in**
radvd                                   **Never logged in**
squid                                   **Never logged in**
mysql                                   **Never logged in**
flenov          ftpd2022 192.168.77.10  Mon Feb 21 12:05:06 +0300 2005
named                                   **Never logged in**
robert          tty1                    Mon Feb 21 12:10:47 +0300 2005
```

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 командной оболочки установлены **/bin/false**, **/dev/null**, **/sbin/nologin** и др.). Если вы заметили, что кто-либо проник 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 этой команды можно определить, какие файлы и какими пользователями открыты в данный момент. Результат выполнения директивы 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 в данный момент вы один работаете с системой, количество открытых файлов может исчисляться парой десятков, и число их заметно растет, если в системе несколько пользователей, ведь один файл может открываться несколько раз каждым из них. Это касается в основном системных конфигурационных файлов.

12.5.2. Системные текстовые журналы

Представленные в этом разделе журналы — это текстовые файлы. Их можно без проблем просматривать такими командами, как `cat`, или любыми текстовыми редакторами.

В файле **/var/log/messages** находится основная информация о заходах пользователей, о неверных авторизациях, остановках и запусках сервисов и многое другое. В один документ все подобные события поместиться не могут, иначе он будет нечитаемым, поэтому в папке **/var/log/** могут находиться файлы с именами **messages.X**, где X — это число.

Этот журнал один из самых главных для любого админа. Если взломщик пытается подобрать пароль, то вы сможете заметить быстрый рост этого файла и появление большого количества записей о неверной авторизации. На рис. 12.1 показан пример содержимого файла.

Следующий журнал расположен в файле **/var/log/secure**. Что в нем такого особенного? Это самый основной журнал, который нужно проверять максимально часто, и каждой записи следует уделять особое внимание. В этом

файле отображается, под каким именем и с какого адреса пользователь вошел в систему. Например, на сервис FTP может подключаться главный бухгалтер. Если вы увидели, что он пользуется сервисом, но при этом журнал показывает чужой IP-адрес, то это может стать поводом для беспокойства.

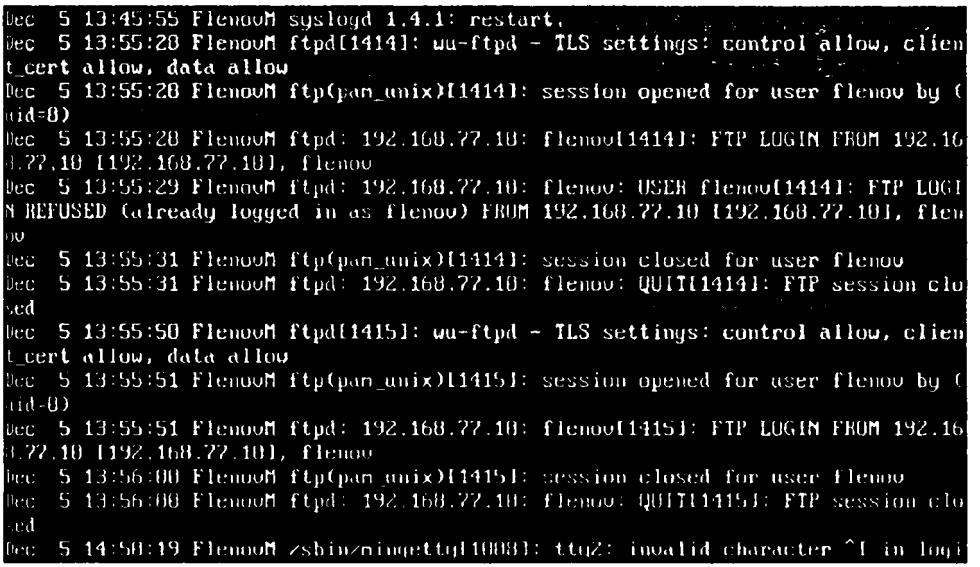

Рис. 12.1. Снимок экрана с открытым файлом **/var/log/messages**

В этом же файле отображается информация о внесении изменений в список пользователей или групп. Злоумышленники очень редко используют запись root для своих злобных целей и заводят какую-нибудь более незаметную, с нулевым идентификатором. Если это сделано не руками, а с помощью команды Linux, то вы в журнале **/var/log/secure** увидите подозрительные изменения.

Хакеры, конечно же, знают о таких уловках, поэтому корректируют список вручную, ведь это не так уж и сложно — добавить по одной записи в файлы **/etc/passwd** и **/etc/shadow**. Но даже в этом случае вы почувствуете неладное, когда увидите запись о том, что в систему вошел пользователь, которого вы не создавали.

Но 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 это отличие не заметишь, и взломщик будет безнаказанно гулять по вашему компьютеру.

Журнал почтового сервера Sendmail находится в файле **/var/log/maillog**. В данном файле можно будет увидеть строки примерно следующего вида:

```
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 осматривал все директории на предмет 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 это не очень надежно. Умные хакеры всегда заметают свои следы и уничтожают ненужные записи из журналов, поэтому ручной осмотр действительно может пригодиться, но первым делом проверьте журнал.

12.5.3. Журнал FTP-сервера

Войдя в систему, взломщики нередко закачивают на сервер собственные программы для повышения своих прав или для открытия потайных дверей. Для закачки можно использовать FTP-протокол. Кто подключался к серверу можно узнать из файла **/var/log/secure**. А вот что закачивали — подскажет **/var/log/xferlog**. О журнале FTP-сервера мы уже немного говорили в *разд. 10.3*. Теперь познакомимся с ним поближе.

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 свой боевой софт на сервере. С помощью анализа журнала вы быстро узнаете, кто и что закачивал.

Посмотрим на пример содержимого файла **/var/log/xferlog**:

```
Sun Jan 16 13:21:28 2005 1 192.168.77.10 46668 /home/flenov/sendmail.cf
b _ o r flenov ftp 0 * c
```

Из данной строки видно, что 16 января в 13:21 пользователь с адреса 192.168.77.10 скачал файл **/home/flenov/sendmail.cf**.

Протокол FTP является наиболее опасным, потому что через него злоумышленник может скачать секретные данные (например, файл с паролями) или

положить на сервер свою программу (в частности, rootkit или трояна). Необходимо научиться понимать каждую запись, чтобы знать, что происходит с файлами в системе. Давайте рассмотрим каждый параметр в журнале:

- полная дата, которая состоит из дня недели, месяца, числа, времени и года;
- продолжительность сеанса или время, потраченное на скачивание/закачивание файла;
- имя или IP-адрес удаленного хоста;
- размер файла в байтах;
- полный путь к файлу, который был скачен или закачен;
- тип передачи — буква `a` (символьная) или `b` (бинарная);
- символ, определяющий специальные действия над файлом:
  - `C` — сжат;
  - `U` — разархивирован;
  - `T` — обработан программой tar;
  - `_` — не было никаких действий;
- символ, определяющий направление передачи: `o` (скачивание с сервера) или `i` (закачивание на сервер);
- символ, определяющий тип пользователя: `a` (анонимный), `g` (гость) или `r` (действительный);
- локальное имя пользователя. Для анонимных пользователей здесь можно увидеть номер ID;
- имя сервиса, обычно это слово `ftp`;
- способ аутентификации. Здесь можно увидеть 0, если определение подлинности отсутствовало, или 1 для идентификации по RFC 931;
- идентификатор пользователя. Если он не определен, то можно увидеть звездочку;
- символ, определяющий состояние передачи: `c` (прошла успешно) или `i` (была прервана).

Если вы никогда не работали с журналом FTP, то советую сейчас остановиться на минуту и внимательно изучить строку примера, показанную выше, и записи из вашего журнала. Вы всегда должны подходить к проблеме уже подготовленными, а не изучать ее после появления, иначе вы проиграете.

12.5.4. 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_REFRESH_HIT` — после обновления выяснилось, что объект в кэше свежий;
  - `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 байт, полученных клиентом;
- метод запроса — `GET`, `POST`, `HEAD` или `ICP_QUIERY`;
- URL-адрес запрашиваемого объекта;
- поле ident (знак "-", если недоступно);
- результат запроса к другим кэшам:
  - `PARENT_HIT` — объект найден;
  - `PARENT_UDP_HIT_OBGECT` — объект найден и возвращен в UDP-запросе;
  - `DIRECT` — объект запрошен с оригинального сервера;
- тип содержимого MIME.

Когда в *гл. 9* мы говорили о squid-прокси-сервере, то упоминали и о других журналах, например, cache.log и useragent.log.

12.5.5. Журнал Web-сервера

Сервер Apache хранит свои журналы в файлах access.log и error.log, которые расположены в директории **/var/log/httpd**. Эти журналы позволяют узнать об активности и доступе пользователей.

С другой стороны, журналы текстовые и легко читаемые. Любой хакер может просмотреть их. А т. к. в журнале сохраняются пароли, с которыми пользователи получили доступ к серверу, то хакер легко их может вычислить.

Отказаться совсем от ведения журнала нельзя. Но необходимо сделать все возможное, чтобы он не был доступен злоумышленнику. Как минимум, я всегда изменяю расположение журнала по умолчанию. По моим наблюдениям редко кто смотрит файл httpd.conf, а лезут сразу в каталоги по умолчанию. Если журналов нет, то хакер думает, что они отключены.

Итак, в директории **/var/log/httpd** создайте пустые файлы access_log, access_log.1, access_log.2, access_log.3, access_log.4, error_log, error_log.1, error_log.2, error_log.3 и error_log.4. Для большей правдоподобности в них можно поместить копию реальных данных, только убедитесь, что там нет важной информации. Правда по дате изменения и по строкам внутри файла злоумышленник легко увидит, что данные старые, но не догадается, что эта информация только для отвода глаз. Главное, чтобы даты изменения файла и формирования записей в нем совпадали.

Для упрощения создания подобных файлов можно на время включить журналы в директории по умолчанию, чтобы они набрали информации, а потом отключить.

После этого измените директивы ErrorLog и CustomLog в файле конфигурации Apache-сервера httpd.conf, указав другую директорию для хранения журналов. Вот такой простой метод позволяет затуманить мозги большинству взломщиков.

12.5.6. Кто пишет?

Записью в журналы, находящиеся в директории **/var/log**, занимаются демоны syslogd и klogd, но в программе setup при настройке автоматически загружаемых сервисов вы увидите только первый. Настройки автозапуска syslogd влияют и на загрузку klogd. Если вы используете изолированную систему или просто не нуждаетесь в протоколировании событий, происходящих в системе, то можете отключить эти сервисы, чтобы они не расходовали процессор-

ное время. Для сервера я не рекомендую этого делать. Если сейчас вы еще не почувствовали необходимость использования журналов, то после первой проблемы или взлома системы увидите 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 параметров 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
```

Назначение директив легко можно проследить по их комментарию. Все они имеют вид:

```
название.уровень
```

В качестве названия выступает имя параметра, который надо протоколировать. Это могут быть следующие категории сообщений:

- `kern` — от ядра;
- `auth` — о нарушении безопасности и авторизации;
- `authprith` — об использовании привилегированного доступа;
- `mail` — от почтовых программ;
- `cron` — от планировщиков задач `cron` и `at`;
- `deamon` — генерируются сервисами;
- `user` — от пользовательских программ;
- `uucp` — UUCP-сообщения (Unix To Unix Copy, копирование с Unix на Unix). В настоящее время практически не используется;
- `news` — из новостей;
- `lpr` — поступает с принтеров.

Уровень может быть один из следующих:

- * — протоколировать все сообщения системы;
- debug — отладочная информация;
- info — информационные сообщения;
- notice — уведомления;
- warn — предупреждения;
- err — ошибки;
- crit — критические сообщения;
- alert — требуется немедленное вмешательство;
- emerg — авария, дальнейшая работа невозможна.

В журнал попадают сообщения указанного уровня и выше. Например, установка значения `err` определяет, что в журнал будут попадать сообщения уровней err, crit и emerg.

Чем больше ошибок вы сохраняете, тем выше нагрузка на жесткий диск, да и расход ресурсов увеличивается. Для большей эффективности функциониро-

вания системы раздел **/var**, на котором хранятся журналы, лучше всего перенести на отдельный винчестер. Таким образом, запись в журналы и работа ОС сможет происходить параллельно. Но вы должны быть уверенными, что раздел **/var** содержит достаточное количество дискового пространства.

Я уже говорил, что в своих системах убираю журналы из расположения по умолчанию, что усложняет хакеру жизнь. Но этого недостаточно. Опытный взломщик просмотрит конфигурационный файл **/etc/syslog.conf** и найдет новое расположение журналов.

Но мы можем поступить еще умнее, и для этого достаточно штатных средств ОС Linux. В моей системе в сервисе cron установлено задание, по которому каждый час делается резервная копия директории **/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, то можно все сообщения журнала отправлять на специально выделенный компьютер, что будет еще более выгодным. Чтобы хакер смог подправить журнал, ему придется взламывать еще один сервер. А если он используется только для протоколирования и никаких лишних портов на нем не открыто, то взлом может оказаться слишком затруднительным.

Для сброса содержимого журнала по сети в файле **/etc/services** должна быть доступна строка:

```
syslog 514/udp
```

После этого открываем файл **/etc/syslog.conf** и добавляем следующую строку:

```
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. 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 файл **/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. Напоминаю, что все

сценарии хранятся в директории **/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

# Source config
# Загрузка конфигурационного файла
if [ -f /etc/sysconfig/syslog ] ; then
       . /etc/sysconfig/syslog
else
       SYSLOGD_OPTIONS="-m 0"
       KLOGD_OPTIONS="-2"
fi

RETVAL=0

umask 077

start() {
       echo -n $"Starting system logger: "
       daemon syslogd $SYSLOGD_OPTIONS
       RETVAL=$?
       echo
       echo -n $"Starting kernel logger: "
       daemon klogd $KLOGD_OPTIONS
       echo
       [ $RETVAL -eq 0 ] && touch /var/lock/subsys/syslog
       return $RETVAL
}
stop() {
# Команды остановки сервиса
}
rhstatus() {
# Команды вывода состояния
}
restart() {
       stop
```

```
        start
}
...
...
```

Самое интересное спрятано в следующих строчках:

```
if [ -f /etc/sysconfig/syslog ] ; then
   . /etc/sysconfig/syslog
else
   SYSLOGD_OPTIONS="-m 0"
   KLOGD_OPTIONS="-2"
fi
```

Здесь происходит проверка. Если файл **/etc/sysconfig/syslog** существует, то используются опции загрузки из этого файла, иначе явно задаются в строке:

```
SYSLOGD_OPTIONS="-m 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, и вам достаточно внести изменения туда, и не надо будет трогать сценарий запуска **/etc/rc.d/init.d/syslog**.

Использование выделенного сервера для сохранения сообщений из журналов позволяет как скрыть записи, так и сделать 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 ему удалось взломать хотя бы один компьютер в защищенной сети, то хакер может установить программу прослушивания и увидеть все сообщения в журналах.

Вопрос решается достаточно просто, если зашифровать трафик, направив его через туннель SSL. Самый простой вариант — выполнить следующие действия:

- на сервере, отправляющем сообщения, в конфигурационном файле прописываем пересылку сообщений на локальный компьютер:

  ```
  *.*  @localhost
  ```

- все сообщения будут передаваться на локальный компьютер, т. е. самому себе на 514 порт UDP. Чтобы все работало верно, сервер не должен находиться в режиме приема сообщений, т. е. запущен без ключа `-r`. Иначе порт 514 будет занят, а нам это не нужно;

- на 514 порту локального компьютера запускаем stunnel-клиент:

  ```
  stunnel -c -d 127.0.0.1: 514 -r loagserver:1050
  ```

  Все сообщения, полученные на этот порт, будут шифроваться и передаваться на порт 1050 компьютера loagserver. В вашем случае вместо имени loagserver нужно указать адрес вашего сервера;

- на компьютере loagserver создаем stunnel-сервер:

  ```
  stunnel -d 1050 -r 127.0.0.1:514
  ```

После этих действий stunnel-сервер будет получать зашифрованные сообщения на 1050 порт и в расшифрованном виде направлять их на порт 514. На сервере loagserver сервис syslogd должен быть запущен с ключом -r для приема сообщений на 514 порту.

Теперь все сообщения будут передаваться по сети в зашифрованном виде.

12.5.7. logrotate

Чтобы файлы журнала слишком сильно не раздувались, в Linux включена утилита logrotate. Рассмотрим принцип ее работы на примере журнала **/var/log/messages**:

1. Когда размер файла **/var/log/messages** превышает максимально допустимое значение или проходит определенный промежуток времени, содержимое текущего журнала переносится в файл **/var/log/messages.1**, а файл **/var/log/messages** очищается и заполняется заново.
2. В следующий раз, при достижении максимального размера, содержимое файла **/var/log/messages.1** переносится в **/var/log/messages.2**, а из журнала **/var/log/messages** — в **/var/log/messages.1**.

Таким образом, история изменений сохраняется в отдельных файлах, и ее можно всегда просмотреть, при этом сам журнал никогда не превысит определенного размера, и с ним будет удобно работать.

За сохранение истории и перенос данных между файлами отвечает программа logrotate. Рассмотрим ее конфигурационный файл (**/etc/logrotate.conf**), который показан в листинге 12.5.

Листинг 12.5. Файл конфигурации программы /etc/logrotate.conf

```
# see "man logrotate" for details
# Выполните команду man logrotate для получения дополнительной информации

# rotate log files weekly
# Смена файлов происходит еженедельно
weekly
```

```
# keep 4 weeks worth of backlogs
# Будут использоваться 4 файла для сохранения истории
rotate 4

# create new (empty) log files after rotating old ones
# После сохранения журнала создается пустой новый файл
create

# uncomment this if you want your log files compressed
# Уберите комментарий со строки compress,
# чтобы файлы журналов сжимались
#compress

# RPM packages drop log rotation information into this directory
# Пакеты RPM переносят информацию о смене журнала в эту директорию
include /etc/logrotate.d

# no packages own wtmp -- we'll rotate them here
# для журнала /var/log/wtmp задаются собственные правила
/var/log/wtmp {
    monthly
    create 0664 root utmp
    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, то можно изменить это значение на `monthly`, чтобы обновление происходило ежемесячно;
- `rotate` — задает количество файлов, которые будут использоваться для хранения истории. В данном 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, компресс-версия может иметь объем на 70 % (и даже более) меньше.

В начале файла идут описания значений по умолчанию. Затем можно указать специфичные значения для определенных журналов. В данном конфигурационном файле выделен журнал **/var/log/wtmp**:

```
/var/log/wtmp {
    monthly
    create 0664 root utmp
    rotate 1
}
```

В данном 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.

За счет смены журналов мы получаем как удобства, так и недостатки. Например, после проведения атаки хакер может уничтожить свои следы, даже если не имеет непосредственного доступа к файлам журнала. Достаточно только засыпать журнал мусорными сообщениями и превысить максимальный размер, чтобы система четыре раза произвела ротацию.

Пытаться защищать журнал, увеличивая его размер до бесконечности, бесполезно, потому что хакер не будет добавлять записи в log-файл вручную, а воспользуется простой программой на Perl или написанной прямо из командного интерпретатора (Shell). Такая программа чрезвычайно проста. Достаточно только в цикле выполнять команду:

```
logger -p kern.alert "Текст сообщения"
```

С помощью директивы `logger` можно записывать в журнал сообщения. Если организовать бесконечный цикл выполнения этой команды, то система сама уничтожит журнал.

Чтобы данные не исчезали бесследно, можно добавить команду, которая будет отправлять журнал на почтовый адрес администратора:

```
/var/log/wtmp {
    monthly
    size = 100k
    create 0664 root utmp
    postrotate
# Команда отправки журнала имя.1
    endscript
    rotate 1
}
```

В данном случае после первой смены журнала будет выполняться сценарий отправки этого файла на почтовый ящик администратора.

Если вы настраиваете сервис таким образом, убедитесь, что ваш почтовый ящик способен принять необходимое количество данных. Например, если вы установили максимальный размер файла в 10 Мбайт, а ваш почтовый ящик способен принять только 5 Мбайт, то вы никогда не получите этот файл, он будет удален почтовым сервером.

12.5.8. Пользовательские журналы

Все команды, которые выполняются пользователем, сохраняются в файле **.bash_history** (если используется интерпретатор команд **/bin/bash**), который находится в пользовательской домашней директории. Когда вы определили, под какой учетной записью в системе находился взломщик, его действия можно проследить по этому журналу.

Вы сможете узнать, какие команды или программы выполнялись, а эта информация подскажет, как злоумышленник проник в систему и что изменил. Если хакер добавил пользователя или модифицировал какой-либо важный системный файл, то вы это увидите и сможете вернуть все в исходное состояние, и тем самым быстро закрыть двери в системе, которые открыл хакер.

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 очищать. Любой (да и вы сами) может 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 поль-

зовательского журнала. Такая ошибка может вам обеспечить не одну бессонную ночь.

Пароли администраторов могут указываться в командной строке и при использовании программы mysql. Если вы выполнили команду `/usr/bin/mysql -uroot -ppassword`, то она полностью сохранится в журнале. Получив доступ к вашему журналу bash-команд, 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 базы данных MySQL, то в вашей домашней директории помимо файла **.bash_history** будет еще и **.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. БД MySQL — это еще не вся ОС, но может послужить отправной точкой для дальнейшего взлома, к тому же сами базы данных могут содержать секретную информацию, например, пароли доступа к закрытым частям Web-сайта.

12.5.9. 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 вы пропустили момент взлома.

Если стали быстро увеличиваться журналы, значит, возросла активность и, возможно, началась атака отказа от обслуживания. Нужно срочно реагировать, иначе рост нагрузки на сервер или каналы связи может стать неуправляемым, и к определенному моменту сервер перестанет обрабатывать запросы пользователей. К тому же, если журналы заполнят весь диск, то система может выйти из строя. Убедитесь, что у вас достаточно места для хранения файлов журналов.

Компьютер не должен перегружаться без вашего ведома. Если это произошло, то проверьте журналы и выясните, по какой причине и когда это случи-

лось. Момент последней загрузки Linux легко узнать с помощью команды `uptime`.

Следите за повторяющимися записями, особенно об авторизации. Если на каком-либо сервисе происходит много попыток входа с неверной идентификацией пользователя, это может говорить о том, что взломщик пытается подобрать пароль.

Если вы заметили что-то подозрительное (с учетом вышеперечисленного), то следует выяснить, откуда идет потенциальная угроза, а именно IP-адрес и расположение сети атакующего. Для предотвращения дальнейших действий со стороны взломщика можно изменить политику сетевого экрана, добавив запись, запрещающую любые подключения со стороны атакующего хоста.

При анализе журнала нужно быть очень бдительным и обращать внимание на каждую мелочь. Например, чтобы подобрать пароль, злоумышленник может использовать различные методы маскировки, в частности самостоятельно записывать в журнал подложные записи.

Если хакер будет подбирать пароль простым перебором, то вы легко увидите большое количество неудачных попыток войти под определенным пользователем, т. к. при этом в журнале **/var/log/messages** появляется следующая запись:

```
Feb 12 17:31:37 FlenovM login(pam_unix)[1238]: authentication failure;
logname=LOGIN uid=0 euid=0 tty=tty1 ruser= rhost= user=root
```

Параметр `login(pam_unix)` указывает на то, что хакер только пытается войти в систему. Если он уже проник, но неудачно использовал команду `su`, то в поле `logname` будет имя, под которым хакер находится в системе, и текст `login(pam_unix)` будет заменен на `su(pam_unix)`.

По такой строке вы легко сможете определить, что это злоумышленник, и быстро найти его. Но хакер может накидать в журнал записей, которые будут указывать на других пользователей, тогда среди них найти реальную будет очень сложно. Например, с помощью следующей команды хакер может добавить в журнал строку, которая будет идентична ошибке аутентификации:

```
logger -p kern.alert -t 'su(pam_unix)' "authentication failure ;
logname=robert uid=0 euid=0 tty=tty1 ruser= rhost= user=root "
```

В ответ на это в журнале будет создана примерно следующая запись:

```
Feb 12 17:31:37 FlenovM login(pam_unix)[1238]: authentication failure;
logname=robert uid=0 euid=0 tty=tty1 ruser= rhost= user=root
```

Теперь представим, что хакер накидал строк, в которых поле `logname` всегда разное. Тогда выделить из них реальные практически невозможно.

Благо если при использовании директивы `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 будет 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 команда `logger` не будет доступна хакеру, он может создать записи своей программой. Чтобы этого не произошло, журналы должны быть доступны на запись только администратору root.

12.6. 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 Web-сервере ежедневный журнал может превышать 4 Мбайта. Это очень много текстовой информации, в которой быстро найти нужную строку практически невозможно.

Именно поэтому программисты и администраторы написали и продолжают разрабатывать программы-анализаторы файлов. 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 посылается информация от работающего сервера по сети. По мере получения записей они проверяются и помещаются в файлы для дальнейшего хранения и более тщательного анализа. По одной записи очень сложно выявить атаку, лучше видеть динамику событий. Например, одна неправильная

попытка авторизации еще ни о чем не говорит, а вот 10 и более — это уже должно вызывать подозрение.

Самое обидное, что все известные программы неэффективны для анализа в динамике. Большинство из них имеют ограничения при создании правил, по которым отдельная запись относится к разряду опасных. Из-за этого приходится в круг подозреваемых относить все, что имеет ошибки входа в систему, а потом самостоятельно анализировать записи и время их срабатывания. Каждый день хотя бы один пользователь может ввести неправильный пароль, особенно если он сложный. Реагировать на подобные записи бессмысленно.

Есть еще один недостаток просмотра журнала построчно. Допустим, что анализатор выдал сообщение о попытке обращения к запрещенной области диска. Для большинства сервисов в этой записи отсутствует информация о пользователе.

Например, вы заметили строку о неправомерном доступе к FTP-директории. В ней будет IP-адрес клиента, но вы можете захотеть узнать, под какой учетной записью зарегистрировался пользователь. Чтобы это увидеть, необходимо открыть сам журнал и проследить историю подключений с этого IP-адреса. А ведь это можно решить, если анализировать журнал в динамике.

12.6.1. tail

Когда я нахожусь непосредственно за сервером, то в отдельном окне терминала запускаю следующую команду:

```
tail -f /var/log/messages
```

После этого на экране появляется содержимое журнала, которое изменяется в реальном времени. При записи в журнал новой строки она тут же появляется у меня на мониторе.

Это очень удобно, особенно при большом количестве записей. Вы можете спокойно работать в одном терминале и иногда переключаться на другой, чтобы взглянуть на ход работы. Если сообщения появляются слишком часто (с сервером работает большое количество пользователей), то проследить за ними просто невозможно. В этом случае необходимо их фильтровать с помощью специализированных программ, которые отображают только подозрительную информацию, а остальные записи просто пропускают.

12.6.2. Swatch

Это очень мощная утилита, написанная на простом и знакомом многим администраторам языке Perl. Это позволяет легко изменять и добавлять возможности самостоятельно. Скачать программу можно по адресу **http://sourceforge.net/project/swatch**.

Утилита позволяет анализировать записи по расписанию (если занести выполнение программы в cron) или сразу после их попадания в журнал.

Так как это Perl-программа, то процесс ее установки отличается от рассмотренных ранее. В данном 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 медали есть оборотная сторона. То, что программа написана на Perl, является и недостатком, т. к. требует наличия интерпретатора. Я уже говорил, что нельзя устанавливать на сервер то, что может открыть дверь злоумышленнику и при этом не является сверх нужным. Без необходимости я рекомендую не подключать интерпретаторы типа Perl, потому что хакеры очень часто используют их для написания собственных rootkit-программ.

12.6.3. Logsurfer

Одна из немногих программ, которая может просматривать журнал в динамике, — Logsurfer (**http://www.sourceforge.net/projects/logsurfer**). Как я уже говорил, большинство средств разбирает журнал построчно, что не очень эффективно, потому что в результате мы получаем слишком много мусора.

Благодаря мощным возможностям этой программы для анализа журнала, мы получаем более сложную настройку. Это тоже недостаток, потому что из-за ошибок в конфигурации можно не уловить очень важные события.

12.6.4. Logcheck/LogSentry

Самая простая в использовании программа. Она написана теми же программистами, что и PortSentry, которую мы рассматривали в *разд. 12.4.2*. В состав LogSentry уже входят различные шаблоны, с помощью которых можно выделять потенциально опасные записи.

Разобраться с программой очень просто, но меня смущает только ее будущее. В последнее время создается впечатление, что обновлений больше не будет, а рано или поздно возможностей текущей версии просто не хватит, и придется искать новое средство.

Но будем надеяться на лучшее.

12.7. Безопасность журналов

Заканчивая тему журналирования, необходимо поговорить и о безопасности. Журналы создавались как средство наблюдения за системой и выявления атак, но могут быть использованы в корыстных целях.

Рассмотрим классический пример взлома. Система регистрирует в журналах безопасности неверные попытки входа. При этом в файл попадает имя пользователя, который допустил ошибку. Сам пароль при этом не сохраняется, чтобы хакер, прочитав журнал, не смог его увидеть. Допустим, что легальный пользователь случайно вместо имени набрал свой пароль. Такое бывает, особенно по утрам, если человек пришел на работу не выспавшись или просто с плохим настроением. Таким образом, пароль будет сохранен в журнале в открытом виде.

Очень важно сделать так, чтобы журнал не был доступным для злоумышленника. Выполните сейчас следующую директиву для просмотра прав доступа на файлы журналов:

```
ls -al /var/log
```

Результат работы команды:

```
drwxr-xr-x  9 root   root     4096 Jan 12 13:18 .
drwxr-xr-x 21 root    root     4096 Jan 24 23:00 ..
drwx------  2 root   root     4096 Jan 12 11:14 bc1security
-rw-r-----  1 root   root    83307 Jan 12 13:18 boot.log
-rw-r-----  1 root   root    89697 Jan 6 09:01 boot.log.1
-rw-r-----  1 root   root    48922 Jan 30 11:45 boot.log.2
-rw-r-----  1 root   root    64540 Jan 23 19:55 boot.log.3
-rw-r-----  1 root   root    36769 Jan 16 12:36 boot.log.4
-rw-r-----  1 root   root     8453 Jan 12 13:18 cron
-rw-r-----  1 root   root     8507 Jan 6 09:06 cron.1
-rw-r-----  1 root   root     7189 Jan 30 11:50 cron.2
-rw-r-----  1 root   root     6935 Jan 23 20:01 cron.3
-rw-r-----  1 root   root     4176 Jan 16 12:41 cron.4
...
...
```

Владельцем всех файлов должен быть администратор root. Убедитесь также, что только он имеет полные права, а всем остальным не позволено работать с журналами.

По умолчанию для большинства файлов правом чтения обладает владелец и пользователи его группы, в качестве которой, чаще всего, можно увидеть root. Если 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 директории **/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 читать и писать мог только администратор.

Помимо этого, на директорию **/var/log** у пользователей не должно быть права 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 строки 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. Таким образом, можно поставить дополнительную защиту от удаления и изменения с помощью ключей. В файловых системах Ext2 и Ext3 есть команда `chattr`, с помощью которой можно устанавливать дополнительные атрибуты на файлы. 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 произошли месяц, а то и год назад. Перед стиранием устаревшей информации из журнала необходимо снять атрибут:

```
chattr -a /var/log/boot.log
```

Только не забудьте потом вернуть его на родину, чтобы файл снова стал доступным только для добавления записей.

Помимо журналов необходимо защищать и программы, установленные для их анализа. Бесполезно стоять на страже файлов, если их можно просмотреть через утилиты. Для этого у всех программ, позволяющих читать журналы, не должно быть установленного SUID- или SGID-бита.

12.8. Безопасность сети

Обеспечение безопасности одного только сервера неэффективно. Необходимо контролировать работоспособность всей сети. Как минимум, вы должны следить за всеми каналами связи.

Самый простой способ — использование утилиты nmap. Она позволяет выполнять команду `ping` сразу на всю сеть, и по ответам можно узнать, что в данный момент доступно. Если какой-то компьютер не ответил, то необходимо выяснить причину. Возможно, это просто потеря питания или незапланированная перезагрузка. Но если это проделки хакера, то вы должны знать об этом первыми.

Утилита nmap хороша для однократного выполнения, но неудобна для мониторинга. Я больше предпочитаю CyD NET Utils (**http://www.cydsoft.com**). Единственный недостаток этой программы — она работает под Windows ☺. Проблема в том, что под *nix вообще подобных программ нет, а контролировать сеть можно и с клиентского Windows-компьютера. Многие так делают, чтобы работать в графическом режиме, а на сервере использовать текстовый.

Рассмотрим, как работает программа. Запустите ее и создайте новый проект (**File/New project**). Перед вами откроется окно, как на рис. 12.2. Слева находится список различных сетевых устройств: компьютеры, серверы и т. д. В центральной части окна есть рабочее поле, на котором можно проектировать свою сеть, расставляя необходимые элементы.

Выберите необходимый пункт в дереве компонентов и щелкните в рабочей области. Перед вами появится окно, в котором можно ввести данные о пользователе, компьютере и его составляющих (рис. 12.3). Это позволит в дальнейшем наблюдать за сетью и вести учет не только техники, но и всех пользователей.

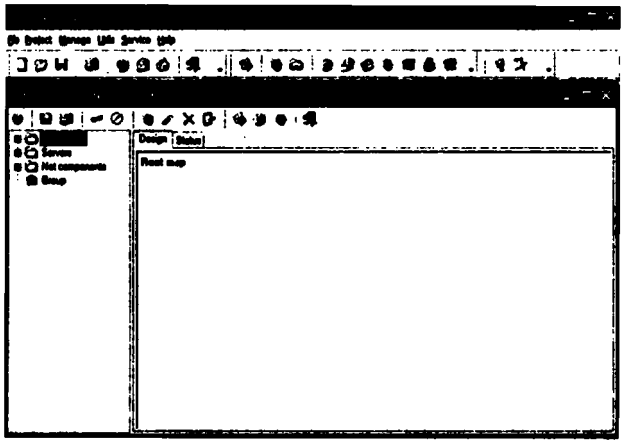

Рис. 12.2. Новый проект в программе CyD NET Utils

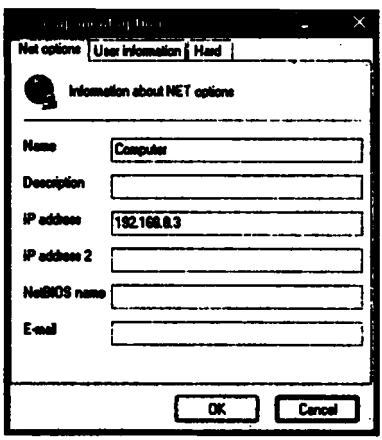

Рис. 12.3. Свойства сетевого компонента

Пункт **Group** позволяет задать группу машин, объединяя в организации целые отделы или просто компьютеры, выполняющие схожие функции, чтобы не загромождать рабочую область. Дважды щелкните мышью по группе, чтобы войти в нее и визуально расставить компоненты. Затем нарисуйте связи, которые будут показывать, как проложен кабель. Для этого нажмите кнопку **Connect two computers** ( ) на панели инструментов (см. рис. 12.2). Теперь щелкните по одному изображению компьютера, связь которого надо указать, а потом по второму. На экране появится пунктирная линия, соединяющая два сетевых устройства.

На рис. 12.4 изображена схема сети из двух компьютеров, сервера и группы, объединенной коммутатором.

Теперь запустим мониторинг сети. Для этого выберите меню **Project/Start**. Программа начнет с определенным интервалом пинговаться ко всем компьютерам вашей сети, для которых на схеме указаны IP-адреса или имена. Интервал задается в настройках программы (меню **Service/Options**).

Если компьютер доступен и команда `ping` выполнена удачно, то рядом с его изображением появится зеленый круг, иначе — красный. Перейдя на вкладку **Status**, можно увидеть описание последнего выполненного сканирования.

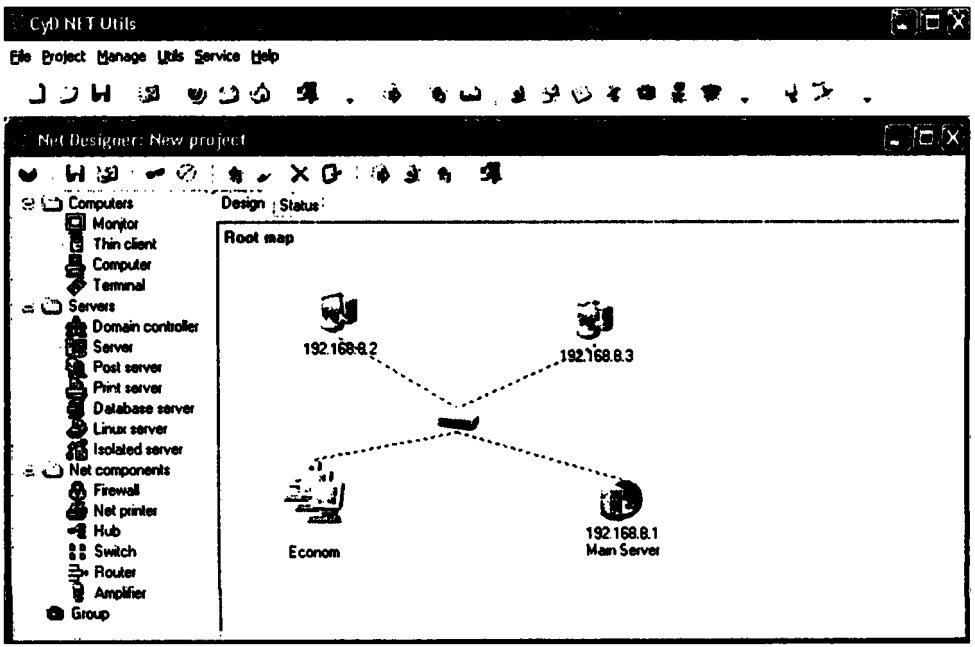

Рис. 12.4. Схема сети, созданная в CyD NET Utils

В пакет CyD NET Utils входит еще одна удобная утилита — CyD Careful Observer, позволяющая следить за работой различных сервисов. Запустите утилиту и выберите меню **Project/Add observer object**. Перед вами откроется окно, как на рис. 12.5.

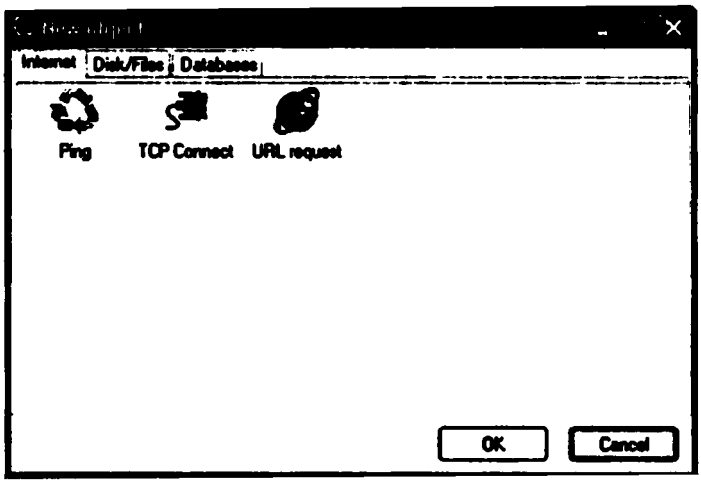

Рис. 12.5. Окно добавлвния нового объекта, за которым необходимо наблюдать

В данном окне можно выбрать следующие объекты наблюдения:

- **Ping** — компьютер, к которому будет направляться ping-запрос через определенные промежутки времени;
- **TCP Connect** — подключение к порту (например, 21), проверка осуществляется через определенные промежутки времени. Если соединения не было, то сервис недоступен;
- **URL request** — URL-адрес. Если он недоступен, значит нет связи с Web-сервером или уничтожен файл.

Остальные возможности больше отражают специфику Windows-систем, а для нас и этого достаточно.

Выбрав объект для наблюдения, нажмите кнопку **OK**. Перед вами откроется окно, в котором можно указать параметры выполняемой операции (рис. 12.6). Например, для утилиты ping можно указать размер пакетов, время ожидания и IP-адрес удаленного компьютера.

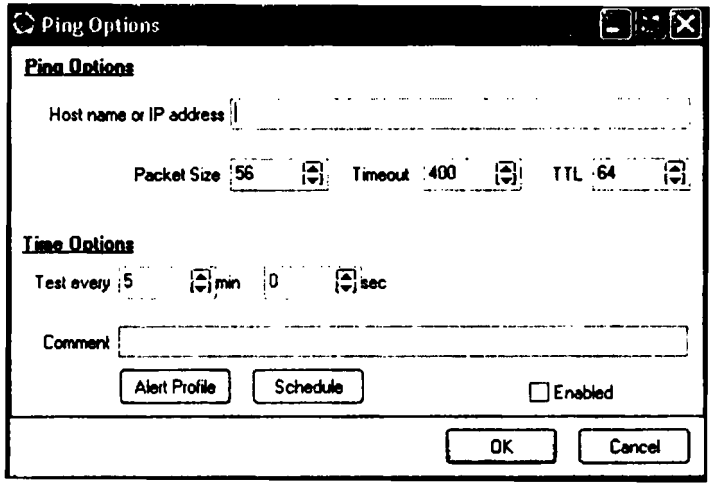

Рис. 12.6. Задание параметров наблюдения

Помимо этого, можно задать время слежения и интервал. Нажав на кнопку **Schedule**, можно установить дни, в которые нужно производить наблюдение. Нажмите кнопку **Alert Profile**, чтобы выбрать (или создать) один из профилей реагирования в случае обнаружения ошибки. Программа может в этом случае выполнять следующие действия:

- направлять сообщение NET SEND (используются в ОС Windows для обмена текстовыми уведомлениями между компьютерами);
- проиграть звук;
- выполнить внешнюю программу или команду;
- отправить E-mail-сообщение;
- просто отобразить на экране окно диалога с ошибкой;
- перезагрузить компьютер (удобно, если наблюдение идет за локальной машиной и найдена критическая ошибка).

В программу встроена хорошая система просмотра журнала происшедших событий. Таким образом, если вы отлучились от компьютера, всегда можно увидеть, что происходило, даже если не включен ни один профиль событий (Alert profile).

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 5 компьютеров, полностью сгорел вместе с кабинетом один сервер. А кто из жителей великих башен города New York думал, что к ним в гости прилетят самолеты? Кто-то скажет, что такие слава безжалостны, ведь произошла трагедия, но мы тоже не застрахованы от жестоких действий террористов, и Россия уже воочию столкнулась с этим. И как бы не 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.

13.1. Основы резервного копирования

Чтобы финансовые потери от утраты информации были минимальными, нужно знать, откуда может прийти угроза. Помимо того, вы должны проанализировать данные, которые вы сохраняете. От этого зависит, как часто нужно производить резервное копирование и каким методом.

Скорость восстановления работы после внештатной ситуации зависит от того, как хорошо вы подготовитесь. Необходимо заранее проиграть все возможные варианты и желательно отработать процесс восстановления на практике, используя тестовую систему, чтобы не пришлось изучать по ходу дела.

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, то очень часто нелепые действия приводят к уничтожению данных. При правильной политике безопасности могут быть разрушены только собственные файлы пользователя, 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 дюймов и жесткие диски с максимальным размером в 20 Мбайт. Если винчестеры были достаточно 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 повышалась. Но когда мы начали оперировать гигабайтами, то в определенный момент я реально столкнулся с проблемой испорченных блоков (Bad Blocks). В определенный период (примерно за полгода) мне пришлось сменить три жестких диска разных производителей размером от 10 до 20 Гбайт. Это было как набег саранчи, которая уничтожала информацию. В настоящее время надежность дисков снова стала улучшаться, 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 конец 2004 и начало 2005 года, то замечаешь, что наша планета начинает преподносить страшные сюрпризы. Я имею в виду участившиеся 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 несгораемом сейфе, то восстановить данные не составляет труда.

- Хакеры и эпидемии вирусов

  Как же без этого. Это чудо информационной жизни, без которого уже никуда не денешься, и приходится выстраивать всевозможные оборонительные рубежи. Какое средство чаще всего используют для защиты от вредоносного кода? Конечно же, антивирусы, но они запрещают выполнение любого известного штама. Но хакеры придумывают новые программы и способы обхода антивирусов. И именно они 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 я надеюсь, что смог вас убедить в необходимости иметь резервную копию всей значимой информации.

К важным данным можно отнести:

- Системные конфигурационные файлы

  На первый взгляд, это не так важно, потому что в таких файлах нет секретной для организации информации. Но конфигурирование ОС и всех программ с чистого листа потребует гораздо больше времени, чем восстановление через резервные настроечные файлы. А это влечет за собой потери из-за простоя, что для некоторых компаний может исчисляться миллионами рублей.

- Документы пользователей

  В директории **/home** могут быть отчетные документы или программы, которые необходимы пользователям для работы.

- Базы данных

  Корпоративные данные хранятся в удобном для работы хранилище — базах данных. Любая компания может понести большие убытки в случае их потери.

- Web-сайты

  Любой динамично развивающийся сайт (от корпоративного до домашней страницы) или портал содержит файлы и сценарии, потеря которых может оказаться ощутимой в денежном эквиваленте.

Базы данных требуют отдельного разговора, потому что в них резервное копирование зависит от средств, предоставляемых СУБД (Система управления базами данных). И этот вопрос мы отдельно затрагивать не будем. Но большая часть теории, рассмотренной в данной главе, может в равной степени относиться как к файлам, так и к базам данных.

13.2. Доступный на все 100 %

Если посмотреть на возможные источники потери данных, то видно, что наиболее вероятными будут вторжения хакера или нарушение работы оборудования. Если в первом случае достаточно восстановить утерянные файлы, то во втором может потребоваться замена оборудования с полной переустановкой системы. Чтобы этот процесс не отнял слишком много времени, лучше всего заранее иметь в наличии комплектующие, которые могут выйти из строя — жесткий диск, память, материнская плата, процессор.

Если в вашей сети каждая минута простоя сервера может оказаться фатальной, то лучше поступить одним из следующих способов: построить кластер или содержать резервные серверы.

Наиболее надежным является построение кластера. Если один сервер выходит из строя, то его нагрузку берет на себя другой. Это позволяет добиться практически 100 % устойчивости оборудования от вероятных неполадок. Но организация кластеров — достаточно сложное и дорогостоящее занятие, поэтому компании стараются найти любые другие возможные варианты.

Большинство программ промышленного назначения уже имеют встроенные средства кластеризации. Их работа достаточно простая и дешевая. В сети находятся несколько серверов, один из которых является мастером (Master), а остальные — подчиненные (Slave). Основной компьютер регулярно посылает в сеть информацию о своей работоспособности и передает подчиненным серверам изменения, которые происходят в базе данных, чтобы на всех машинах была одинаковая копия БД. В том случае, когда связь с главным компьютером прерывается, всю работу на себя берут подчиненные серверы.

Помимо повышения надежности работы кластер может увеличивать и производительность, если все серверы будут функционировать параллельно и нагрузка будет распределена равномерно. Это позволит более эффективно использовать оборудование и пропускную способность сети.

Более дешевым вариантом является использование резервных дисков. Допустим, что у вас есть один сервер, который должен быть всегда доступен пользователям. В нем устанавливаем дисковый массив RAID (Redundant Array of Independent Disks, матрица независимых дисковых накопителей с избыточностью) и обязательно с поддержкой зеркалирования (Mirroring), т. е. RAID 1 или RAID 1+0.

Примечание

RAID 1 — зеркалирование (два диска содержат одну и ту же информацию). RAID 0 — параллельная запись на два диска, за счет чего скорость доступа к устройству увеличивается, но на безопасность это не влияет. RAID 1+0 — комбинация зеркалирования и параллельной записи.

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 есть 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 нештатной ситуации, подготавливаем заранее сервер с идентичной конфигурацией оборудования. В случае нарушения работы железа достаточно подключить RAID-массив к резервному серверу, переключить сетевой кабель и можно продолжать трудиться. Так как оборудование на обоих компьютерах одинаковое, переустановка системы не потребуется, и RAID будет работать на другом. сервере без изменения конфигурационных файлов.

Если в вашей сети несколько серверов с одинаковой конфигурацией, то один резервный компьютер может служить заменой для любого из них. Таким образом, обеспечение сохранности данных становится 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 были подключены большие диски через Mobile Rack (устройство, позволяющее сделать винчестер съемным). 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 на другой компьютер. Для этого у администратора всегда есть подготовленный системный блок, который может заменить испорченное оборудование.

13.3. Хранение резервных копий

Несмотря на использование RAID 1 и кластера, резервное копирование никто не отменял, и его делать необходимо. Но куда производить резервирование? 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 копия делалась на тот же диск, но только в другой раздел. Некоторым людям очень тяжело объяснить, что при поломке винчестера, как правило, становятся недоступными все его разделы, а не один из них.

Но самое интересное во всей этой истории то, что диск начал рассыпаться уже достаточно давно. Так уж получилось, что основной раздел был в начале диска, а раздел для резервной копии — в самом конце. Уже несколько месяцев во время резервирования происходили ошибки доступа, и никто не обращал на это внимания. Диск явно начал сыпаться, начиная с раздела, на котором хранилась резервная копия, и постепенно испорченные блоки покрыли весь винчестер.

Резервную копию всегда нужно делать на отдельный носитель. Это может быть как отдельный жесткий диск, благо цены на них падают не по дням, а по часам, или любой сменный носитель достаточного объема.

Хранение на отдельном носителе позволяет решить проблемы с оборудованием, но не гарантирует защиту от воровства или стихийных бедствий. Меня поражают администраторы, которые используют сейфы для хранения абсо-

лютно бесполезных бумаг, гарантийных талонов, а резервные копии помещают в простой деревянный ящик. Хочется спросить таких людей: "А зачем вы защищаете сервер всеми доступными средствами, когда можно просто украсть копию данных из шкафчика?"

Резервная копия тоже должна быть под надежной охраной. Лучше всего воспользоваться несгораемым сейфом, тогда вашим данным не страшны даже стихийные бедствия.

В настоящее время в Интернете снова начинают расширяться услуги по предоставлению дискового пространства. Сделав резервную копию на сторонний носитель, можно быть уверенным, что данные будут надежно спрятаны. Владельцы сервисов защищают такие диски с помощью RAID-массивов, поэтому на сервере информация не исчезнет.

Можно с уверенностью сказать, что такой подход к хранению данных будет развиваться, потому что компания Apple с помощью своей новой технологии iDisk сделала интернет-диски удобными и доступными для своих и Windows-пользователей. На очереди и остальные системы. Подробнее о технологии iDisk можно узнать на сайте компании Apple **http://www.mac.com/1/iTour/tour_idisk.html** (рис. 13.1).

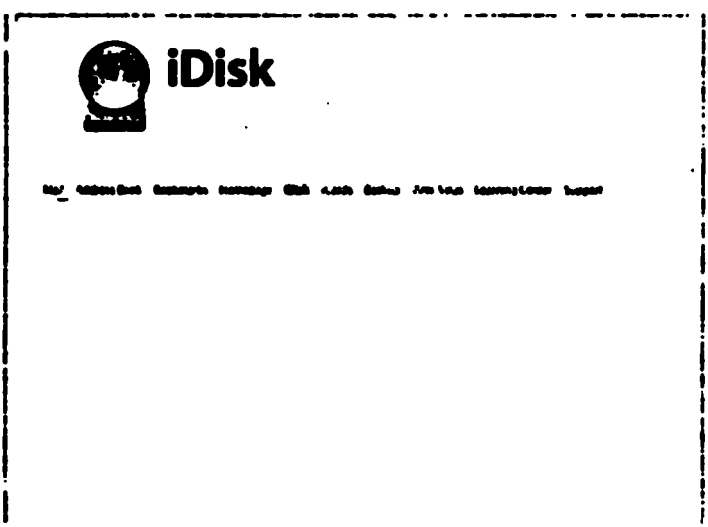

Рис. 13.1. Сайт, посвященный технологии iDisk от компании Apple

Если для вас использование подобного диска слишком дорого (например, неприемлемы затраты на трафик из-за большого количества данных), то о сохранности резервных копий придется позаботиться самостоятельно.

В качестве носителей информации в настоящее время можно использовать съемные жесткие диски, магнитные ленты, CD-R/RW, DVD-R/RW, диски JAZ, ZIP. Выбор конкретного носителя зависит от размера данных и необходимой скорости резервирования.

В настоящее время появились очень удобные внешние винчестеры, подключаемые по USB или FireWire. Такие диски легко переносимы и имеют большой размер. В домашних условиях я использую именно такой носитель, сбрасывая все данные с ноутбука на жесткий диск.

13.4. Политика резервирования

От того, как вы будете резервировать данные, зависит скорость проведения операции и потери после восстановления. Если информация на сервере занимает сотни гигабайт, то необходимо достаточно много времени на ее копирование, что вызовет большую нагрузку процессора. Если процедура выполняется по сети, то и канал связи будет перегружен, что сделает сервер менее доступным.

Ваша задача организовать резервирование максимально эффективным методом, чтобы оно занимало как можно меньше времени, и при этом создавалась копия всех необходимых данных.

При планировании резервирования вы должны учитывать, что если произойдет поломка жесткого диска, то все изменения, внесенные с момента создания последней копии, будут потеряны. В связи с этим необходимо сохранять важные данные как можно чаще, но при этом не забывать, что это достаточно накладный процесс для сервера.

Итак, сколько носителей информации нам понадобится, с какой частотой и как их использовать? Это зависит от многих факторов:

- хранящаяся информация;
- частота изменения данных;
- наличие возможности ручного восстановления большого количества потерянных данных;
- максимальное время простоя (недоступности) сервера;
- категория наиболее часто меняющихся данных.

И этот список можно продолжить, но мы на этом пока остановимся. А просто начнем рассмотрение. Нужно четко себе представлять, какие данные в систе-

ме изменяются. После этого разделите их на три группы в зависимости от периодичности модификации: часто, редко и с определенным интервалом.

Основные директории, которые должны резервироваться:

- **/etc** — содержит конфигурационные файлы;
- **/home** — пользовательские файлы;
- директория, содержащая Web-файлы.

В остальных каталогах администраторы редко держат документы или файлы, требующие дополнительного копирования. Программы из директории **/bin** или **/usr** нет смысла дублировать, потому что их легко переустановить, особенно если сохранена вся конфигурация.

13.4.1. Редко, но метко

К нечасто изменяемым файлам можно сразу отнести конфигурационные файлы (директория **/etc**). В этом каталоге массовые корректировки происходят на этапе установки сервера. Затем компьютер может работать годами, и изменения происходят 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 быть возможность перезаписи. Я для этих целей использую ZIP- и JAZ-диски. В заархивированном виде достаточно одной дискеты.

Так как конфигурация изменяется редко, то можно делать копии сразу после внесения правок. Для этого достаточно записать измененный файл на диск без копирования всех конфигурационных файлов.

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.

Воссоздание прав может произойти неверно, особенно нужно уделить этому внимание, если вы применяете дополнительные средства фильтрации разрешений, используя программы, предоставляемые сторонними разработчиками.

Прежде чем восстановленную систему сделать доступной, необходимо убедиться, что все файлы находятся в том же состоянии, как перед сбоем, особенно права доступа.

13.4.2. Зачастили

Часто изменяемыми могут быть базы данных и основные файлы и документы пользователей (директория **/home**), которые корректируются каждый день. Их резервные копии можно и нужно создавать ежедневно. Если процесс ко-

пирования отнимает слишком много времени, то следует это делать после 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).

Для этих данных я использую 7 перезаписываемых носителей. Каждый из них я называю соответственно дням недели, потому что в понедельник копирую информацию на диск с надписью "Понедельник", во вторник пишу на диск "Вторник" и т. д. Помимо этого, каждый понедельник все данные записываются на одноразовый носитель типа CD-R или DVD-R.

13.4.3. Часто, но не все

Далеко не все файлы в директории **/home** изменяются ежедневно. Большинство из них не трогается годами. Чтобы не тратить каждый раз время на такие данные, можно использовать команды, которые позволят копировать только то, что корректировалось. Самый простой вариант — выбрать все файлы, у которых дата изменения находится в определенном промежутке времени.

При использовании такой политики можно действовать следующим образом:

- в конце недели производится полное копирование директории **/home**;
- каждый день можно сохранять измененные файлы.

В случае аварии восстановление должно происходить точно в той последовательности, в которой происходило резервирование. Сначала воссоздается полная копия. Потом по очереди возвращаем на место все файлы из резервных копий. Если порядок будет нарушен, то вы рискуете заместить новый файл более старым.

Копирование данных по дате изменения удобно, но доступно не всегда. В большинстве утилит есть только обновление существующей копии. В этом случае сначала создается полная копия, а потом с помощью специального ключа задается обновление файлов, которые были изменены.

Этот способ хорош, но он заменяет все старые файлы. После этого нельзя откатиться назад и узнать, что было до последнего резервного копирования. С другой стороны, при наличии полной копии для восстановления достаточно скопировать ее в систему, и работа может продолжаться.

Каждый день изменяется не так уж много файлов, поэтому резервирование будет происходить достаточно быстро, и его можно делать в процессе работы сервера. Но в данном случае вы рискуете испортить документы. Допустим, что есть два файла, информация в которых жестко связана. Если во время ко-

пирования одного файла другой будет модифицирован, то в резервную копию первый попадает измененным, а второй — нет. После восстановления могут возникнуть серьезные проблемы в работе, потому что нарушится целостность данных.

13.4.4. Периодично

Данные, которые изменяются с определенным интервалом, нужно резервировать в соответствии с этим параметром. Например, некоторые файлы используются во время ежемесячной отчетности. Как правило, они достаточно большого размера, и создавать регулярно 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 в независимости 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:

- необходимо много времени;
- слишком большая нагрузка на сервер;
- невозможно реализовать средствами Linux. В большинстве дистрибутивов нет необходимых программ;
- в резервную копию попадают все файлы, и даже те, которые не являются необходимыми, например директория **/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.

13.4.6. Носители

Теперь рассмотрим, сколько носителей нам понадобится для хранения всех резервных копий. Для каждого типа данных нужны свои носители, потому

что их копирование происходит с разной периодичностью и рассматривать их надо отдельно:

- *конфигурационные файлы*. Мы уже определились, что для них достаточно ZIP- или JAZ-диска. Желательно иметь две одинаковых копии, потому что любые дискеты портятся и выходят из строя намного чаще, чем жесткие диски;
- *периодично обновляемые данные*. Их лучше всего записывать на съемный носитель и хранить длительное время. Я для этих целей использую CD-R, потому что его объема достаточно для моих данных и при этом информацию нельзя стереть. Каждый месяц я делаю такой диск и сохраняю его в течение года. Таким образом, я по резервной копии всегда могу просмотреть данные за любой отчетный период;
- *часто обновляемые данные*. При выборе носителя главным критерием должна быть скорость работы, потому что чаще всего эти данные имеют большой размер. Резервное копирование должно производиться максимально короткое время, чтобы сервер не ощутил долговременных нагрузок.

Как видите, политика резервирования зависит от многих факторов. Я постарался показать вам основные принципы, по которым вы должны строить свою схему. Предложенный в этом разделе подход не сможет одинаково подойти для всех систем, но его можно использовать как базовый.

13.5. Резервирование в Linux

Мы будем рассматривать 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 есть встроенного 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 для ОС, который будет выполнять все необходимое.

13.5.1. Копирование

Самый простой вариант создания резервной копии — использование команды `cp` (копирование файлов). Только в этом случае необходимо обязательно сохранять права доступа к файлу. Вот как может выглядеть команда, дубли-

рующая директорию **/home** на примонтированном к системе диске **/mnt /resdisk**:

```
cp -a /home /mnt/resdisk
```

В данном случае я использовал ключ `-a`, который равносилен указанию сразу трех ключей (`-dpR`):

- `-d` — не следовать по символьным ссылкам. Мы копируем каталог в таком виде, как он есть;
- `-p` — сохранять права доступа к файлу;
- `-R` — копировать каталоги рекурсивно, чтобы на архивный диск попали все файлы и подкаталоги.

Получается, что команда, показанная выше, идентична следующей:

```
cp -dpR /home /mnt/resdisk
```

С помощью этой директивы мы создаем полную копию каталога. А как можно сохранить изменения? Для этого необходимо произвести копирование на тот же диск, только с указанием ключа `-u`:

```
cp -au /home /mnt/resdisk
```

Таким образом будут скопированы все файлы из директории **/home**, дата изменения которых позже документов из директории **/mnt/resdisk**.

13.5.2. tar

Копирование по одному файлу не очень удобно. Лучше, когда все находится под одной крышей. В Linux есть утилита, которая собирает все файлы в один. Этот процесс называют архивированием, но вы должны учитывать, что tar не сжимает файлы. Если вы объединяете файлы общим размером в 2 Мбайта, то архив будет иметь размер чуть больше (размер всех файлов плюс заголовок tar).

Преимущество сборки целой директории в один файл состоит в том, что им проще потом управлять и сжимать специализированными программами.

Для архивирования, как минимум, нужно выполнить команду:

```
tar cf архив.tar директория
```

Здесь у нас два параметра:

- `c` — указывает на необходимость создания архива;
- `f` — назначает архивный файл или устройство (по умолчанию используется **/dev/rmt0**).

Итак, для архивирования директории **/home** выполняем следующую команду:

```
tar cf backup.tar /home
```

При использовании параметров `cf` файлы в архив попадают с сохранением пути. Если распаковать созданный нами чуть выше архив, то в результате в текущем каталоге будет создана папка **/home**, а в ней уже будут располагаться все пользовательские директории. Например, если разархивировать из директории **/home**, то папки пользователей окажутся в **/home/home**, а если находится в **/etc**, то пользователи увидят свои директории в **/etc/home**.

Чтобы добиться правильного результата, нужно перейти в корень диска и выполнять команду там:

```
cd /
tar xf /home/backup.tar
```

В данном случае 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 отличия между архивом и файлом в системе;
- `t` — просмотреть содержимое архива;
- `u` — обновить файлы в архиве, которые были изменены;
- `n date` — добавить в архив только те файлы, которые изменены позже даты, указанной в параметре `date`.
- `P` — не удалять первый символ "/". В этом случае, откуда бы вы не распаковывали, файлы попадут на свое родное место.

С помощью утилиты tar можно архивировать сразу несколько директорий. Следующая команда помещает в архив **/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 директории 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 текущей директории.

13.5.3. gzip

В ОС Linux есть достаточно много различных утилит для упаковки данных. Наиболее популярной из них является gzip. Преимущество архивирования над простым копированием данных заключается в том, что результирующая копия занимает меньше места, а значит, носитель для резервирования нужен меньшего объема.

Чаще всего резервированию подлежат документы, размер которых в заархивированном виде может уменьшаться на 90 %. Текстовые данные сжимаются намного лучше, чем программы.

Недостаток архивирования — возрастает нагрузка на процессор и может потребоваться больше времени на создание полной копии.

За счет того, что архив занимает намного меньше места, его копирование на сетевые ресурсы или запись на съемные носители (ZIP, JAZ, CD-R/RW, DVD-R/RW и др.) будет производиться быстрее. В итоге может получиться, что временные затраты на архивирование (с учетом занятости процессора) будут равны времени на копирование без архивирования.

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 директории **/home**, применяя наибольшую компрессию. Выполните следующую команду:

```
gzip -9 backup.tar
```

Теперь просмотрите содержимое директории (команда `ls`). Обратите внимание, что файла backup.tar больше нет. Вместо него появился 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 xfz /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 можете увидеть файл backup.tar, а backup.tar.gz исчезнет.

Теперь вы готовы написать свой сценарий, который будет собирать директории для архивирования в tar-файл, а затем сжимать его, чтобы уменьшить его размер. Но зачем использовать две команды, когда можно обойтись одной. Вот как это делается:

```
tar cvf - /home | gzip -9c > backup.tar.gz
```

В данном примере мы собираем в tar-архив директорию **/home** и тут же сжимаем ее утилитой gzip.

Помимо gzip для архивирования иногда используется утилита compress, но ее возможности по сжатию ниже, и к тому же вокруг нее были 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. dump

Все предыдущие команды, которые мы рассматривали в данной главе, не являются специализированными командами резервирования. Это просто команды копирования и архивирование файлов. Утилита dump предназначено именно для создания резервной копии файловой системы Ext2.

Для выполнения резервной копии нужно, как минимум, указать:

- `-n` — уровень резервной копии, который может изменяться от 0 до 9. При значении `0` создается полная резервная копия. Уровни выше 0 означают формирование резервной копии изменений, произошедших с момента последней полной резервной копии или создания копии с меньшим уровнем;
- `-u` — требование при удачном завершении резервирования обновить файл **/etc/dumpdates**, в котором хранятся даты резервных копий;
- `-f файл` — имя файла или устройство, на которое нужно производить резервное копирование.

Итак, простейшая команда создания полной резервной копии выглядит следующим образом:

```
dump -0u -f /home/backup.bak
```

Для сохранения изменений нужно изменить уровень, указав значение больше нулевого, например:

```
dump -1u -f /home/backup.bak
```

Для восстановления файлов из архива используется команда `restore`. Но прежде чем ее запускать, вы должны убедиться, что находитесь в директории, которая принадлежит восстанавливаемой файловой системе.

Директиве `restore` достаточно только указать ключ `-f` и файл, который нужно восстановить. Если применить ключ `-i`, то вы попадаете в интерактивный режим, в котором можно задать файлы для восстановления. Интерактивный режим похож на командную строку, в которой можно путешествовать по архиву и выполнять следующие директивы:

- `help` — вывести краткую помощь по доступным командам;
- `ls` — отобразить содержимое текущей директории;
- `pwd` — показать текущую директорию;
- `add` *директория* — добавить в список для восстановления указанный в качестве аргумента каталог;
- `cd` — сменить текущую директорию;
- `add` *директория* — удалить из списка восстановления директорию, указанную в качестве параметра;
- `extract` — восстановить все файлы из списка;
- `quit` — выход.

13.6. Защита резервных копий

Нет смысла защищать систему, если компакт-диски с резервными копиями беспорядочно лежат у вас на столе. Резервные копии хранят все основные данные компьютера, и если диск попадет в руки хакера, то уже не надо будет ничего взламывать.

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 уже можно разархивировать все файлы на свои места.

ГЛАВА 14

Советы хакера

В этой главе нам предстоит познакомиться с различными атаками и методами взлома, которые могут использовать злоумышленники. Чтобы защитить систему, вы должны знать, как в нее могут проникнуть, а чтобы взломать, нужно быть в курсе, как обороняться. Это правило действует всегда и во всем, а не только в компьютерном мире. Как защититься от вора, если не иметь представления, как он взламывает замок? Если знать, что он станет его пилить, то можно сделать все возможное, чтобы этот процесс отнял слишком много времени, и вора можно было успеть остановить. В этой главе нам предстоит познакомиться с методами компьютерных преступников, чтобы пополнить свой арсенал противоядием.

Некоторые из рассматриваемых вопросов будут носить общий характер, потому что всякий способ имеет слишком много вариаций, и не всегда можно дать четкие рекомендации. Например, атака вирусов. Вроде бы все просто — есть зловредный код, который нужно искать и уничтожать. Но вирусы бывают разные, и для каждого из них могут быть свои методы обезвреживания. В то же время можно сформулировать общие правила для борьбы с ними. Пусть эти советы не дадут 100 % результат, но, по крайней мере, помогут в битве с врагом.

Опытные пользователи или администраторы могут заметить, что какие-то рекомендации устарели. Но кто не знает крылатое выражение "Все новое — это хорошо забытое старое". В теперешней сети очень много людей, которые присоединились к процессу совсем недавно. Они знакомы с современными технологиями, но при этом не знают истории. Я стал замечать, что хакеры снова стали использовать методы, которые работали 10 или 20 лет назад, и делают это успешно.

Почему давние методы взлома успешно воплощаются в жизнь? Опытные пользователи, которые имеют опыт борьбы и знают разные способы вторже-

ния, просто со временем забывают про эту опасность, а новые — еще не обожглись.

При том количестве серверов и людей, которое насчитывает современная сеть Интернет, обязательно найдется хотя бы 1000 человек, которые попадутся на простейших методах взлома. Это связано с тем, что уровень подготовки пользователей Интернета невысок. Я не имею в виду школьную программу, я говорю об образовании в сфере компьютерной безопасности. Никто не обучает простых пользователей, а администраторы либо ленятся, либо просто не хотят тратить деньги на повышение квалификации.

14.1. Основы безопасности

Мы начинали книгу с рассмотрения того, как взламываются компьютеры, а напоследок поговорим про общие принципы безопасности. Некоторые аспекты, которые мы будем рассматривать, затронут только ОС Linux, а кое-какие советы можно применять к любой операционной системе и компьютеру (серверу) в целом.

В этой главе я развею некоторые мифы о безопасности и покажу множество примеров из собственного опыта.

Почему необходимо уметь строить оборонительные рубежи? Неужели ОС или серверные программы не могут быть изначально защищенными? Вы сами должны помнить об этом постоянно и именно потому, что ОС и программы всегда уязвимы, т. е. имеют погрешности, позволяющие хакеру получить доступ к файлам или возможностям, которых у него быть не должно.

Лазейки есть всегда, потому что программы пишут люди, а им свойственно ошибаться. Получается, что даже в самой защищенной программе есть дыра, просто ее, наверное, еще не нашли. Спросите любого хакера про самое защищенное ядро Linux, и он вам скажет, что последняя версия отличная и не содержит багов. Повторите вопрос через месяц, но на этот раз тоже ядро окажется дырявым и без заплаток, и с ним работать не рекомендуется.

При появлении каждой новой версии ОС мы слышим, как она надежна и безопасна, но через короткое время появляются обновления. Избежать ошибок невозможно, и с этим надо смириться и регулярно устанавливать исправления.

В большинстве случаев я бы не назвал уязвимости ошибками, потому что программы работают верно, просто хакер использует их особым образом и выводит из строя систему. Такие ситуации предусмотреть очень сложно, и задача разработчика максимально их сократить.

Итак, рассмотрим основные принципы безопасности, которые могут понадобиться в нашей повседневной жизни.

14.1.1. Ответственность

Первое, с чем необходимо разобраться, — кто должен отвечать за безопасность системы. В большинстве организаций этим занимаются администраторы, и это первая ошибка. Человек, который настраивает систему, может не иметь исчерпывающей информации по безопасности и просто не будет видеть своих ошибок.

В администрировании очень часто возникают классические ошибки обмана зрения. Я написал достаточно много работ и постоянно сталкиваюсь с такой проблемой. Когда вы читаете свой текст, то видите его таким, какой он должен быть, а не то, что есть на самом деле. Например, вы можете написать слово "горад", а читать его как "город" только потому, что так было задумано.

Ошибка в одной букве конфигурационного файла может привести к плачевным последствиям, и администратор ее может не увидеть, потому что воспринимает параметр таким, как надо. Зато любой взгляд со стороны без проблем зацепится за ошибку. Лучше всего, если таким наблюдателем будет специалист.

Администратор должен настраивать и обслуживать систему, а специалист по безопасности обязан проверять настройки и тестировать систему на уязвимости. Конечно же, оба человека должны помогать друг другу и могут быть взаимозаменяемы, но не стоит их объединять в одном лице, особенно в крупных компаниях.

Оплата высококвалифицированных специалистов должна быть достойная, и здесь не стоит экономить, потому что человеческий фактор — самая большая дыра в безопасности.

14.1.2. Защищайте только то, что нужно

Многие специалисты по безопасности рекомендуют защищать только то, что нужно. Действительно, зачем охранять мусорную корзину, когда в ней находятся одни только отходы, которые никому не нужны. Тут же вспоминается знаменитый фильм "Хакеры", в котором главные герои залезали в мусорный бак. Что они там искали? Различные документы или бумажки. Нередко пользователи записывают пароль на листочках, или им раздают параметры доступа в виде распечаток, именно это искали хакеры.

То же самое и с файловой системой. Некоторые безобидные папки могут оказаться хорошим материалом для хакера. Однажды я тестировал на безопасность систему, в которой была открыта только одна директория с файлами, содержащими тексты песен группы "Dune". Вроде бы невинная вещь, и что можно сделать через текстовые файлы?

Я запустил подбор пароля для пользователя root и в качестве словаря указал эти файлы. Каково же было мое удивление, когда я увидел, что установленный пароль root — название группы "Dune". Взлом занял всего несколько секунд!!!

Очень часто в открытые папки администраторы выкладывают информацию, связанную с их интересами, и если параметры доступа тоже заданы исходя из этих пристрастий, то и на подбор пароля потребуется не более 5 минут.

С другой стороны, получив хоть какой-то доступ к системе (особенно на запись данных), у хакера появляется возможность повысить свои привилегии аж до администратора. В Интернете все чаще появляются сплоиты, которые позволяют это сделать. Если злоумышленник вообще не имеет доступа к системе, то и взломать ее сложнее.

На данный момент известно не так уж много способов для удаленного взлома, зато, имея локальный доступ к системе, вероятность повысить свои права увеличивается в несколько раз. Защищать компьютер от проникновения по сети проще, и основным способом в этом случае является использование сетевого экрана. Если хакеру все же удалось пробраться, то тут уже все зависит от правильной политики регламентации доступа. Если хоть где-то есть ошибка, то хакер сможет получить права администратора.

Когда злоумышленник пытается удаленно пробиться в систему, защищенную сетевым экраном, то его возможности сильно ограничены еще из-за того, что он слишком мало знает об атакуемой системе. Проникнув внутрь, объем информации сразу увеличивается в несколько раз.

Основными целями, которые подвергаются атакам хакеров, являются:

- *уязвимые программы ОС*. Если посмотреть отчеты по безопасности, то вы увидите, что дыры в различных утилитах появляются почти каждую неделю, и программисты с администраторами только и успевают их затыкать;
- *программы сторонних разработчиков*. Все программное обеспечение, которое включается в дистрибутив производителем, тщательно тестируется. Сторонние же разработчики осуществляют проверку только на своем дистрибутиве, и нет гарантии, что программа будет также безопасно и стабильно работать со всеми вариантами ОС Linux. К тому же профессионализм некоторых "чужих" разработок оставляет желать лучшего, и мы об этом уже говорили, когда рассматривали безопасность открытого программного обеспечения *(см. разд. 1.3)*;
- *сценарии и программы, написанные администратором системы или программистами вашей организации*. Очень часто для расширения возможностей ОС пишутся собственные сценарии (чаще всего используется интерпретатор Perl), и нередко именно они становятся причиной взлома. Только профессионал, хорошо знакомый с принципами безопасности и правилами

создания кода, сможет создать защищенный сценарий или программу. Любители и простые администраторы слишком мало внимания уделяют различным проверкам параметров, чем незамедлительно воспользуется хакер.

Итак, разделение на "важно" и "неважно" должно быть. Значимые данные должны защищаться лучше, их надо шифровать и устанавливать за ними более тщательное наблюдение. Но при этом необходимо думать о системе в целом.

Были случаи, когда администраторы один сервер с закрытой информацией защищали и любили, а другой — открывали для всеобщего обозрения и использования. Это правильное решение, но между этими серверами не должно быть доверительных отношений, и имена пользователей и пароли следует делать разными. Но т. к. мы ленивые, то пароль root для всех серверов, чаще всего, один и тот же или отличается незначительно, чтобы легко было запомнить. Если не совершать этой ошибки, то решение с физически разделенными для разных задач серверами является верным решением.

14.1.3. Нет поблажек

Допустим, что вы администрируете сервер на крупном предприятии с большим количеством офисов. Это самое сложное с психологической точки зрения. Почему? Сейчас увидите.

Большинство администраторов защищают сеть от внешнего вмешательства. Но по статистике большинство взломов, причем самых жестоких, происходят внутри системы (работниками фирмы, друзьями и т. д.), потому что администраторы не выдерживают натиска друзей и коллег, требующих дать пароль или доступ к определенному ресурсу. Вы не должны поддаваться на такие уговоры. Если предоставить большие привилегии другу, он может наказать вас взломом. Пусть он будет и случайным или просто шалостью, но ликвидация последствий может оказаться достаточно проблематичной.

Безопасность — это не только надежная защита системы от взлома, но и устойчивость к неправильным действиям пользователя. Хотите пример? Их у меня много, но простейший и чаще всего встречающийся — это эффект начальника. Многие администраторы считают, что руководитель или директор фирмы должен иметь все права, чтобы иметь возможность просмотреть любую информацию. Идея хорошая, но реализуется достаточно сложно, потому что начальники, получив доступ на чтение, начинают просить право на запись, чтобы можно было что-то изменить. А это уже намного страшнее, особенно, если шеф не IBM-совместимый и плохо знает компьютер. Именно такие просят максимальные права, и неумелые/случайные действия могут уничтожить все данные. Виноватым сделают вас.

Еще один недостаток повышенных прав для друзей и начальства — невозможность защитить их пароли. Когда только один пользователь (root) имеет максимальные права в системе, то его пароль защитить достаточно просто. Но если таких 10 человек, то за их паролями уследить сложнее. Любой из этих пользователей может выбрать себе слабый по стойкости пароль или просто записать его на бумаге. Благодаря этому хакер может получить привилегированный доступ к системе и натворить бед.

14.1.4. Защита рабочего места

Охрана рабочего места является не менее важной, чем защита ОС и ее сервисов. Когда мне пришлось работать программистом в большой компании, в мои обязанности входили: разработка модулей сбора информации с производственного оборудования, настройка и установка сервера с программой в цех и сопровождение ПО. Для каждого компьютера я выбрал разные пароли, которые были сложными для запоминания.

Я хорошо позаботился о безопасности, но через некоторое время, когда пришел снимать копию базы данных на случай краха системы, увидел на мониторе написанный маркером пароль. Возникает вопрос — зачем я мучался и придумывал хитрую комбинацию, когда любой проходящий мимо работник фирмы или даже посторонний мог ее увидеть?

Начинающие пользователи не любят запоминать сложных вещей, поэтому рисуют пароли на мониторе, клавиатуре или на листиках, которые лежат прямо на столе, тогда все попытки обезопасить систему бессмысленны.

Защищать рабочее место нужно так же, как и доступ по сети. Из моей практики могу сказать, что большинство взломов происходят именно из-за безответственного отношения самих пользователей.

Для предохранения рабочего места следуйте следующим рекомендациям:

1. Никогда не записывайте пароли на листиках и других заметных и легко доступных местах. Потратьте немного времени, чтобы запомнить основные кодовые комбинации.
2. Отходя от компьютера, блокируйте клавиатуру утилитами vlock или xlock, или выходите из системы, чтобы в ваше отсутствие никто и ничего не мог делать.
3. Если хакер получил доступ к клавиатуре работающего компьютера, поменяйте пароль текущего пользователя — это дело пяти секунд. Никогда не надейтесь на хранитель экрана, потому что пока он включится, вы потеряете контроль над своей учетной записью. Я отключаю хранитель экрана, чтобы его наличие не смущало меня, а вместо этого выработал привычку всегда блокировать работу компьютера.

4. Если вы используете графический режим, то никогда не держите ярлыки на рабочем столе. На экране могут быть видны только значки по умолчанию, а если поместить туда ссылку на чужой компьютер, с которым вы работаете, то хакер увидит это. Такая информация может сильно развязать руки злоумышленнику.
5. Необходимо разрешать вход в BIOS (Basic Input Output System, базовая система ввода вывода) только с помощью пароля. Если хакер получил физический доступ к компьютеру, то он сможет его перезагрузить в однопользовательском режиме и далее взломать пароль root.
6. Защищайте паролем загрузку ОС в конфигурационном файле LILO. Проблемы загрузчика мы рассматривали в *разд. 3.2*.
7. Перезагрузка компьютера не должна быть случайной или незапланированной. Отключите в файле **/etc/inittab** строку, отвечающую за сочетание клавиш <Ctr>+<Alt>+<Del>.

14.1.5. Документация по безопасности

Многие считают документацию уделом бюрократов и категорически не пишут никаких инструкций или других руководств. Я сам до поры до времени был таким и предпочитал следить за системой самостоятельно, пока она не стала слишком сложной, что не охватить одному, и не произошел взлом.

Мы рассмотрим много рекомендаций по обеспечению безопасности, и впоследствии многие правила станут для вас родными, — вы будете знать, как действовать во время и после взлома. Но пользователи останутся в неведении.

Рассмотрим простейший пример. Хакер проник в систему и получил исключительные права root. Вы закрываете уязвимость и меняете пароль администратора, но хакер возвращается в систему. Почему? Во время взлома хакер мог украсть файлы **/etc/passwd** и **/etc/shadow** и расшифровать пароль (подобрать с помощью специализированных утилит для хранящихся в файле зашифрованных его версий). После взлома все пользователи должны поменять свои пароли. Для решения этого вопроса есть два способа:

1. Самостоятельно сгенерировать пароли и раздать пользователям. В большой организации это удобно, потому что можно быть уверенным, что все пароли изменены, но с передачей могут быть проблемы.
2. Написать инструкцию по безопасности, которая обяжет всех пользователей по первому сигналу администратора сменить свои пароли. Этот документ должен быть прочитан всеми сотрудниками.

Из своей практики могу посоветовать использовать комбинацию этих двух методов: пользователи сами должны сменить пароли, но если этого не про-

изошло в течение 3 часов после сигнала, то пароль меняется автоматически. В этом случае уже пользователи будут бегать к вам, чтобы узнать, как войти в систему.

Помимо этого, в инструкции по безопасности должны быть рекомендации по составлению сложного и длинного пароля, а вы просто контролируйте выполнение.

Через документацию вы можете заставить работников различных служб помогать вам в обеспечении безопасности. Например, администраторы серверов не могут отследить, когда увольняется кто-то из сотрудников. А ведь после ухода уже бывший служащий может отдать свои параметры доступа другому человеку, и он проникнет в систему.

Случаи со взломами, совершенные бывшими сотрудниками, случаются достаточно часто. Я сам неоднократно столкнулся с этим, когда администратор был уволен, но никто не изменил пароли. Через пару дней все файлы на сервере были уничтожены. Я в это время работал программистом в той фирме, и все сотрудники отдела ИТ в срочном порядке восстанавливали данные.

14.1.6. Пароли

Не буду повторяться, что пароли должны быть сложными, а поговорим немного о другом.

Все пароли необходимо менять через определенные периоды времени. На своем сайте (а к нему имеют доступ достаточно много человек) я это делаю каждый месяц, потому что в Интернете слишком много хакеров, которые только и ищут легкую добычу.

На серверы я также меняю пароли каждый месяц, а на особо важные — даже каждую неделю. Это сложно, потому что нужно постоянно запоминать новые комбинации, зато более безопасно.

Единственное, что не меняется, — это пароль на вход в Windows на моем ноутбуке. Со своим компьютером я работаю один и никого не подпускаю. Не потому, что боюсь обнародовать что-то важное, а больше опасаюсь случайной потери данных.

Многие хакеры, получая доступ к системе, некоторое время не проявляют никакой активности. Они осматриваются, знакомятся с принципами работы системы и определяются, как сделать, чтобы их не вычислили. Быстрые действия можно ожидать только от того, кто проникает в систему ради уничтожения всех данных, и кому нет смысла уничтожать следы своего пребывания. Слава богу, что 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 оче-

редной смены злоумышленник теряет свои права, и требуется повторный взлом для выявления нового пароля.

Регулярное обновление паролей усложняет 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, даст результат, если пароль несложный и постоянный. Если пароль изменчив, то вероятность успеть его подобрать до очередной 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 тупым 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.

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? Для этого есть утилита chage, которая имеет следующий вид:

```
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. Меньше 3 дней нежелательно, потому что существуют выходные, и если срок действия попадет на воскресенье, пользователь не успеет сменить пароль. По умолчанию стоит значение –1, что соответствует отсутствию проверки;
- `-M N` — максимальный диапазон в днях (`N`), в течение которого действует пароль. После этого параметры авторизации считаются недействительными, 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 дней и не более 4 дней, чтобы приостановить действие записи на время отпуска или болезни работника;
- `-W N` — количество дней до окончания срока действия пароля, когда пользователю будет выводиться предупредительное сообщение. Нежелательно указывать менее 3 дней, чтобы не попасть на выходные;
- `-l пользователь` — информация о времени жизни их пароля. С этим параметром команда может вызываться любыми пользователями. Чтобы получить сведения о пароле root, выполните директиву `chage -l root`.

  Результат выполнения команды имеет следующий вид:

```
Minimum:      -1
Maximum:      99999
Warning:      -1
Inactive:     -1
Last Change:    Feb 04, 2004
Password Expires: Never
Password Inactive:Never
Account Expires: Never
```

  Здесь отображаются следующие значения:

  - `Minimum` — минимальный срок действия пароля;
  - `Maximum` — максимальный период для пользования паролем;
  - `Warning` — количество дней, за которые будет выдаваться предупреждение о завершении срока действия пароля;
  - `Inactive` — максимальное количество дней, в течение которых учетная запись может не активироваться;
  - `Last Change` — последняя дата изменения;
  - `Password Expires` — дата окончания действия пароля;
  - `Password Inactive` — дата, когда пароль стал неактивным;
  - `Account Expires` — дата окончания действия учетной записи.

Чтобы задать максимальное количество дней жизни пароля в 60 дней, выполните команду:

```
chage -M 60 robert
```

Слишком частая смена паролей приводит к тому, что пользователи просто не успевают запомнить их. Из-за этого сложные комбинации начинают писать на бумаге, чтобы случайно не забыть, или просто меняют пароль на старый. Ваша задача — контролировать замену, и в то же время не стоит заставлять пользователя делать это слишком часто. Период в 2—3 месяца (или 60—90 дней) считается вполне приемлемым.

А какие гарантии, что пароль выбран сложный или пользователь снова не указал старый шифр? В этом нам поможет PAM-модуль pam_cracklib.so, который выполняет основные проверки. Например, нельзя будет установить старый пароль или воспользоваться большей его частью.

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
```

В этой команде мы заставляем систему использовать библиотеку pam_cracklib.so. Параметр `retry` задает 5 попыток для ввода нового пароля, а они понадобятся, если пользователь попытается задать слишком простую комбинацию. Параметр `minlength` задает минимальную длину пароля.

14.1.7. BugTraq

Честно сказать, настоящих хакеров в мире не так уж и много. Большинство взломов совершается подростками, которым нечего делать и хочется где-то применить свои силы. Знаний у них не бог весть сколько, поэтому в основном используются уже готовые решения, найденные настоящими хакерами. Это значит, что вы должны следить за новыми методами взлома и всеми появляющимися уязвимостями. Я для этого использую сайт **www.securityfocus.com**. Здесь регулярно обновляется информация по этим вопросам и предлагаются способы защиты.

Уже давно ходят споры о том, нужны ли сайты наподобие **www.securityfocus.com**. С одной стороны, они позволяют администраторам получать сведения об уязвимостях, а с другой — хакеры узнают, как можно взломать систему. Я считаю, что такие сайты должны существовать, проблема тут не в этом. Просто большинство администраторов никогда сюда не заходят, а узнают о наличии "тонких мест" в программном обеспечении только тогда, когда их сеть/сайт/сервер взломаны. Даже если вспомнить какую-либо брешь девяностых годов, можно найти в Интернете компьютер или сервер, который содержит эту уязвимость. Я бы таких администраторов увольнял без разговора.

Если вы думаете, что ежедневные проверки обновлений смогут спасти систему, то сильно ошибаетесь. После того, как найдена уязвимость, и до момента выхода обновления проходит некоторое время, и в этот период опасность проникновения на компьютер максимальная. Любой хакер, узнавший метод взлома, может начать штурм, и обязательно добьется успеха. Вы должны раньше него узнать об уязвимости и принять меры по предотвращению вторжения, пока не появится обновление.

Уязвимыми оказываются как сервисы, так и ядро ОС. Если погрешность найдена в программе, то ее нужно обновить, скачав более новую версию с сайта производителя. С ошибками в ОС дело обстоит сложнее. Тут необходимо обновление ядра, что не так уж и просто, если производить операцию из исходных кодов. Но можно облегчить себе задачу, если использовать RPM-пакет, тогда установка будет не намного сложнее, чем любой другой программы.

14.1.8. Патчинг ядра

Помимо официальных обновлений ядра существует множество заплаток, написанных сторонними разработчиками (SELinux, lcap, LIDS и т. д.). Все они предназначены для защиты системы на уровне ядра ОС. Например, можно запретить выполнение кода из стека, что сделает невозможным многие атаки, использующие его переполнение. Некоторые заплатки запрещают просмотр каталога **/proc**, следят за процессами в системе, защищают от сканирования портов и многое другое.

Почему я не рассматривал примеры сторонних патчей ядра более подробно раньше? Большинство из них требуют перекомпиляции ядра и работают не со всеми версиями/ядрами ОС Linux, а также требуют немалых усилий при установке. Безопасность повышается, и это факт, но стабильность может пошатнуться, потому что эти заплатки делаются сторонними разработчиками, а Red Hat просто может их не учесть.

Именно поэтому я не включал обзор этой тему в данную книгу. Но вы должны знать о существовании таких патчей, и может быть возможности какой-то заплатки вам покажутся удобными и помогут в обеспечении безопасности системы. Но любая их установка будет происходить на ваш страх и риск. Вы также должны учитывать, что обновление ядра может вызвать проблемы. К тому же, каждое новое устанавливаемое вами ядро тоже нужно будет патчить.

14.1.9. Развитие

Одна из важнейших составляющих эффективного управления — образование. В России проблема повышения квалификации стоит особо остро. Сколь-

ко администраторов самоучек? Очень много!!! Да и обучение (особенно в регионах) оставляет желать лучшего.

Многие специалисты в сфере безопасности не имеют специального образования. Я достаточно часто общаюсь с администраторами, и, глядя на компьютер, могу сказать, хороший он или нет. Если на компьютере стоят игры, то на 90 % такой администратор свободное время тратит на разборки с монстрами. Если игрушек нет, и софт связан только с профессиональной деятельностью, то такой администратор хороший или может стать таким.

Компьютерные технологии стремительно развиваются, и если свободное время на работе тратить на бег с пистолетом по темным коридорам, то не сегодня, так завтра знания устареют. Нужно постоянно повышать свою квалификацию и осваивать что-то новое.

Специальное образование — это хорошо, но оно дает только базовые знания, которые можно получить из литературы в течение месяца. Более конкретные сведения устаревают еще до того момента, как вы закончите высшее учебное заведение, и без подпитки свежими данными можно превратиться в простого продвинутого пользователя.

Кто хорошо работает, тот должен обязательно отдыхать. Но только надо помнить, что деятельность специалиста по безопасности заключается не только в решении текущих задач, но и в повышении квалификации.

14.2. Переполнение буфера

Это одна из самых популярных и в то же время наиболее сложная в использовании уязвимость. Для начала определимся, почему программисты допускают такие ошибки, при которых возможно выполнить переполнение буфера?

В таких языках, как C++, для работы с данными, которые вводит пользователь, выделяется память определенного размера. Информация заносится в этот буфер простым копированием. Большинство программистов рассчитывают максимальный объем данных, который может передать пользователь, и выделяют именно столько памяти (возможен небольшой запас). При этом во время копирования не происходит никаких проверок размера полученных данных. Злоумышленник может воспользоваться этим недостатком и ввести в программу столько информации, что она просто не поместится в памяти и произойдет сбой.

Откуда берется ошибка? Не буду вас загружать программированием и машинными кодами, а попробуем рассмотреть простейший случай переполнения. Структура программы может выглядеть примерно следующим образом:

```
Код
Код
Буфер данных 50 байт
Код
Код
```

Буфер для хранения данных находится внутри кода. В данном случае будем считать, что программист выделил под него 50 байт. А что будет, если пользователь передаст в программу 70 байт? Если при приеме информации программа не проверит размер блока, то при записи данных в буфер они выйдут за его пределы и запишутся поверх кода. Это значит, что программа будет испорчена и не сможет выполнять каких-либо действий, и, скорее всего, произойдет зависание.

В старых версиях Windows некоторые ошибки переполнения буфера могли нарушить работу всей ОС. Windows 2000/XP и Linux более защищены, их вывести из строя уже намного сложнее, и при подобных ошибках перестает функционировать только сама программа.

Но это еще полбеды. Наиболее опытные взломщики могут передать такие данные, в которых будет 50 байт мусора, а потом пойдет корректно исполняемый код, но только написанный злоумышленником. Тогда структура программы будет выглядеть примерно таким образом:

```
Код
Код
Буфер данных 50 байт.
Код хакера
Код хакера
```

В этом случае код хакера может натворить уже много серьезных дел. Если программа с внедренным кодом работает под правами root, то взломщик может прочитать пароли, открыть какую-либо дверь в системе или просто уничтожить все данные на сервере.

В последнее время ошибок с переполнением буфера становится все меньше, потому что появились средства автоматической проверки кода, но все же их достаточное количество. К тому же, в мире существует не так много хакеров, способных через ошибку переполнения буфера подставить свой собственный код. Но остается возможность написать программку, которая использует ошибку и будет простой в применении. В этом случае любой студент сможет ею воспользоваться для взлома вашего сервера, и это уже будет настоящей угрозой.

Помимо нарушения стека через переполнение буфера хакер может испортить код программы, используя неверное форматирование. Некоторые функции в программировании могут оказаться небезопасными при определенном ис-

пользовании. Взломщик имеет возможность ввести такую информацию, которая при обработке программой изменит ее код. Принцип борьбы с такими ошибками схож с переполнением буфера, поэтому я не буду заострять на этой проблеме внимание, тем более что нас как пользователей и администраторов не волнуют тонкости создания машинного кода программы.

Если вы узнали о том, что один из сервисов подвержен атаке переполнения буфера, и можете временно обойтись без него, то его следует отключить. Если сервис не нужен, то, не задумываясь ни на одну секунду, удалите его с компьютера.

Если сервис необходим в вашей работе, то первым делом обратитесь на сайт разработчика. Там могут оказаться рекомендации по устранению ошибки, выполните их. Бывает, что разработчики предлагают всего лишь подправить какие-либо конфигурационные файлы, а иногда приходится скачивать последнее обновление программы.

Не устану напоминать, что своевременное обновление программ позволяет значительно повысить надежность сервера. В 90 % случаев ошибки переполнения буфера устраняются именно таким образом, потому что связаны с неверной работой кода, и для их исправления требуется вмешательство в логику программы (и перекомпиляция) со стороны разработчика.

При отсутствии рекомендаций по устранению ошибки примите все меры для ограничения прав программы. Если она принадлежит пользователю root и установлен бит SUID или SGID, т. е. программа работает от имени владельца root, даже если ее запустил гость, то необходимо сбросить этот бит.

В качестве самостоятельной защиты от ошибок работы с буфером могу посоветовать воспользоваться утилитой Libsafe (**www.research.avayalabs.com /project/libsafe**). Это библиотека создает промежуточный уровень между программой и ОС, перехватывая опасные системные функции, вызываемые кодом взломщика, заменяя их своими аналогами.

У библиотеки один недостаток — происходит небольшая потеря производительности. Но это ничто по сравнению с преимуществами. Утилита не занимается защитой определенной программы или конкретной ошибки. Она направлена на решение большинства потенциальных проблем. От всего защититься невозможно, потому что хакеры всегда придумывают что-то новое. Но даже если обезопасить систему от части возможных ошибок, то система сможет работать безотказно продолжительное время.

14.3. Rootkit

Проникнув в систему, хакер стремится укрепиться в ней и получить максимальные возможности. Например, он уже может выполнять на сервере команды от имени простого пользователя. Этого ему будет мало, поэтому

следующая цель — получение максимальных 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, — называются rootkit (набор администратора). После этого взломщик выполняет команды следующим образом:

- от имени простого пользователя, правами которого обладает хакер, директивы посылаются программе rootkit;
- программа rootkit выполняет полученные команды от имени администратора.

А как же rootkit получает возможность исполнять переданные ей команды с правами root? В этом помогает злополучный SGID-бит. Если он установлен, то программа будет выполняться в системе с максимальными правами администратора.

Но не все так просто. Для rootkit нужно установить SGID-бит и владельцем определить 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, которая уже имеет установленный SUID- или SGID-бит.

Вот почему в *разд. 12.2* мы так усердно вычищали все SUID- и SGID-программы, каждая из них — это дыра в безопасности, но иногда без этой прорехи жить невозможно. Вам необходимо регулярно следить за такими программами, и в случае появления удалять из системы. Вы также должны держать под контролем все изменения, которые происходят с уже установленными SUID- и SGID-программами. Если их размер изменился, то следует бить тревогу и восстанавливать исходное состояние программы.

Вы должны быть внимательны, когда проверяете SGID-программы. Хакеры знают, что администраторы стараются свести количество таких программ к минимуму, поэтому идут на разные уловки. Например, они могут создать файл **/mnt/mount** с SUID-битом. Программа mount действительно требует этого бита, но должна находиться в директории **/bin**. Если вы просматриваете список найденных SUID-программ бегло, то можете не заметить отличие в пути или вообще не обратить на это внимания.

Помимо этого, в названиях программ может идти игра букв. Например, **/bin/login** не требует такого бита. Хакер может создать файл **/bin/1ogin** (первая буква заменена на цифру 1), и визуально программа действительно долж-

на быть в системе, хотя и без SUID- и SGID-бита, но вы не заподозрите ее в злодеянии.

Пакеты rootkit не ограничиваются только предоставлением доступа к выполнению команд от имени пользователя root. Они могут включать еще и различные вспомогательные утилиты, такие как анализаторы сетевого трафика (sniffer), программы управления файлами журналов, позволяющие чистить следы пребывания хакера в системе, и другие полезные взломщику средства.

Загрузив и установив набор rootkit, хакер закрепляется в системе и впоследствии сможет вернуться, даже если была закрыта уязвимость, через которую он изначально проник. Вы должны уметь находить и уничтожать пакеты rootkit, чтобы преградить путь хакеру, который может раньше вас узнать о следующей дыре в системе.

Для облегчения задач администраторов добрыми людьми была разработана программа chkrootkit. Ее можно найти на сайте **http://www.chkrootkit.org**. На данный момент она способна обнаружить более 50 известных пакетов rootkit. Таким образом, вы без особых усилий можете отыскать и уничтожить в системе потайную дверь для хакера.

Но, как говорится, на бога надейся, а сам не плошай. Готовыми наборами rootkit пользуются только начинающие хакеры или любители. Профессиональный взломщик хорошо знаком с программированием и создаст себе инструмент самостоятельно. Тем более что это не так уж сложно, достаточно знать особенности работы ОС Linux. Поэтому вы должны научиться находить и удалять rootkit.

Определить появление rootkit-пакета вручную поможет сканирование портов. Чтобы воспользоваться потайной дверью, нужно открыть в системе порт, на котором rootkit ожидает соединение со стороны хакера. Взломщик подключается к этому каналу и управляет системой.

Для быстрого сканирования лучше всего подходит пакет nmap (**www.insecure.org**). Это один из самых быстрых сканеров под Linux с большими возможностями. Необходимо запустить программу проверки всех 65 535 портов. Для этого нужно выполнить команду:

```
nmap -p 1-65535 localhost
```

Параметр `-p` позволяет задать диапазон портов. В данном случае установлен весь диапазон от 1 до 65 535.

Помимо этого, может пригодиться один из следующих параметров:

- `-sT` — стандартное сканирование с установкой TCP-соединения, является самым медленным. Любая программа антисканирования увидит его *(см. разд. 12.4)*. Если вы запускаете утилиту nmap под обычным пользователем, то по умолчанию будет применяться этот метод;

- `-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 канал 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 закрыт. А вот работа Windows далека от стандарта, и здесь результат не предсказуем;
- `-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 ответить ошибками;
- `-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 данный 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, chkrootkit включает в себя еще и утилиту ifpromisk, которая позволяет найти программу прослушивания трафика.

И напоследок, проверяем список работающих процессов с помощью команды `ps -aux` на предмет наличия незнакомых процессов. При просмотре будьте также внимательны. Вспомните пример с программой login, когда первая буква "l" заменялась на цифру 1. Едва бросив взгляд, можно не заметить процесс 1ogin.

Если объединить работу всех этих утилит в одно целое, то можно будет получить новый пакет rootkit, о котором еще неизвестно фирме chkrootkit.

После того как вы определили наличие файлов rootkit, вы должны остановить их работу и удалить из системы. Самое простое, если программа хакера не модифицировала никаких системных файлов. Если 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
```

В данном случае мы запрашиваем восстановление пакета с именем `пакет.rpm`.

14.4. backdoor

Если взломщик получил доступ к серверу, то, чтобы оставаться незаметным, он устанавливает в системе программы backdoor (потайные двери). Такие программы чаще всего действуют следующим образом:

- на каком-либо порту открывается прослушивание, и программа ожидает подключения хакера;
- когда соединение состоялось, то программа открывает для хакера командную оболочку на этом порту, чтобы можно было выполнять директивы.

Это вам ничего не напоминает? Да, троянские программы работают подобным образом, но троянов подбрасывают как вирусы и ожидают, что администратор сам их запустит, а backdoor взломщик закачивает на сервер и запускает сам.

Есть сходство и с программами rootkit. В настоящее время стирается грань между разными хакерскими утилитами. Одна программа может выполнять сразу несколько функций. Так rootkit и backdoor уже давно соединяют в одно целое, хотя остаются еще и классические утилиты.

Надо уточнить, что программы backdoor нельзя купить в ближайшем магазине. Взломщики пишут их для собственного использования. И все же на закрытых сайтах можно встретить некоторые из этих разработок.

Хакеры не очень любят раскрывать свои программы, потому что если они станут достоянием общественности, то лазейки, через которые взломщик проникает в систему, закроют.

Так как основа этой книги — организация безопасной системы, то я не стану рассматривать процесс создания и открытия потайных дверей. Мы будем обсуждать проблему их поиска и уничтожения.

Самый простой и быстрый способ найти чужую программу — просмотреть процессы, работающие в системе, и открытые порты. Как следует из определения, backdoor — это программа, которая ожидает подключения взломщика, а значит, должен присутствовать работающий процесс этой программы. Выполняем команду `ps` и смотрим, что сейчас работает в системе. После этого сканируем открытые порты, используя утилиту netstat, и ищем злостную программу.

При просмотре процессов нужно убедиться, что файл программы ps не модифицирован хакером. Так как исходные коды ОС Linux доступны, злоумышленник может изменить программу ps так, чтобы она не отображала процесс backdoor, и подбросить свой вариант в вашу систему.

Доступность исходных кодов позволяет хакеру изменять и любые другие программы. Например, может быть трансформирован демон `telnetd`, и после этого, помимо основных своих функций, программа будет играть роль потайного входа. Убедитесь, что исполняемые файлы всех работающих процессов не изменены.

Некоторые демоны могут работать с подгружаемыми модулями. Злоумышленник может написать и подключить свой модуль вместо или в дополнение к стандартным, и его определить уже сложнее, т. к. основной процесс не модифицирован.

Изменение исходных кодов достаточно сложное занятие, и нужно обладать хорошими знаниями в программировании, поэтому данный метод является наиболее опасным, но мало распространен. И все же, его нельзя сбрасывать со счетов, потому что никогда не знаешь, какой квалификации взломщик проник в систему.

При просмотре процессов будьте внимательны. Хакер может назвать свою утилиту telnetd, и тогда в вашей системе будет две программы с таким названием. Одна будет системной, а другая — хакерская, которая выполняет функции backdoor. Будьте бдительны при просмотре списка открытых процессов.

Если ваш сервер работает постоянно, то хакер может смело запускать свой процесс backdoor и уходить восвояси. Если сервер хоть иногда выключается, то злоумышленник должен позаботиться о том, чтобы после перезагрузки backdoor тоже запустился, иначе потайной вход в систему будет закрыт.

Обязательно проверьте все сценарии, отвечающие за загрузку сервисов, на предмет изменений. Они находятся в директории **/etc/rc.d/init.d**. Это может быть сложно, потому что в ОС Linux таких сценариев достаточно много. Но хакер в любой из них может добавить команды загрузки своей утилиты.

Допустим, что вы не нашли никаких лишних процессов, и ни один сервис или программа не изменены. Это еще не значит, что потайного входа нет. Недавно я встретил новый способ прятать процессы от недремлющего ока администратора — модули ядра.

С недавних пор ядро Linux стало действительно модульным. Это удобно, потому что позволяет получить новые возможности, просто подгрузив необходимый блок. *Если раньше для этого требовалась* перекомпиляция ядра, то теперь всего несколько команд, и все готово.

Как же взломщики используют ядро, чтобы спрятать свой процесс? Программа ps (и подобные ей) для определения запущенных процессов использует ядро. Именно оно знает, что работает в данный момент. Хакерами были написаны разнообразные модули, которые не дают ядру сообщить об определенных процессах, поэтому администратор просто не увидит программу backdoor. Как раз поэтому продолжаем анализировать систему в поисках злопрограмм.

Итак, помимо запущенного процесса, программа backdoor должна открыть какой-то порт и ожидать подключения со стороны хакера. Таким образом, мы должны контролировать и это.

Самый быстрый способ определить сервисы, ожидающие подключения — это использовать команду `netstat`. Но т. к. одноименная 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.

Лучший способ спрятать backdoor от сетевых анализаторов — использовать при программировании RAW Sockets (сырые сокеты), как это делают снифферы. На сервере программа backdoor прослушивает весь трафик, и если видит пакеты, помеченные специальным образом, то выполняет инструкции, описанные в этом пакете. 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 снифферы, поэтому они тут бессильны. О снифферах мы поговорим в следующем разделе, а сейчас я только скажу, что для прослушивания трафика сетевая карта должна работать в специализированном режиме, который легко определяется, если просмотреть состояние сетевого интерфейса командой `ifconfig`.

Единственный недостаток — во время работы сниффера повышается нагрузка на сервер. В этом случае все пакеты, которые проходят мимо сетевой карты, поднимаются до уровня ОС.

Но даже спрятанную программу backdoor можно найти. В данном случае можно поступить по правилу "клин клином вышибают". Запускаем сниффер и просматриваем, что проходит через нашу сетевую карту. Если мы видим пакеты, которые отсылают закрытую информацию или пароли, то это может указывать на наличие в системе программы backdoor. От сниффера может скрыться только зашифрованный трафик.

Но самый лучший способ защиты от backdoor — хорошо настроенный сетевой экран. Если в применяемой вами политике безопасности по умолчанию все запрещено, а разрешен только доступ к публичным ресурсам, то даже если сторонняя программа откроет какой-то порт, то подключиться к нему бу-

дет невозможно без изменения фильтров в сетевом экране. Следите за тем, чтобы никакие лишние записи там не появлялись, и все мучения хакера станут напрасными.

14.5. Подслушивание трафика

Мы уже говорили в *разд. 1.4.4* о том, что в локальных сетях очень популярным методом взлома являются программы снифферы, т. е. подслушивание трафика. В Интернете такие программы тоже можно использовать, но здесь задача усложняется, хотя и осуществима. В данном разделе мы рассмотрим теорию реализации атаки с использованием сниффера и поговорим о том, как можно определить его наличие в сети.

Как мы уже знаем, снифферы перехватывают пакеты, которые адресуются другим компьютерам. Так как большинство протоколов создавались на заре становления Интернета и не используют шифрования, то в проходящих пакетах можно увидеть в открытом виде пароли доступа, номера кредитных карт и другую приватную информацию.

Изначально программа сниффер создавалась как помощник в работе администраторов. Но хакеры нашли ей другое применение, и простой анализатор работы сети превратился в мощное оружие взломщика.

Снифферы могут работать в активном или пассивном режиме. В *разд. 14.5.2 и 14.5.3* нам предстоит познакомиться с обоими типами, но чтобы наш разговор был более понятным, вам необходимо знать, что такое модель OSI (Open Systems Interconnection, взаимодействие открытых систем).

14.5.1. Open Systems Interconnection

Каждый раз, когда вы передаете данные по сети, они как-то перетекают от вашего компьютера к серверу или другому компьютеру. Как это происходит? Вы, наверное, скажете, что по специальному сетевому протоколу, и будете правы. Но существует множество их разновидностей. Какой и когда используется? Зачем они нужны? Как они работают? Вот на эти вопросы я и постараюсь ответить в этом разделе.

Прежде чем разбираться с протоколами, нам необходимо узнать, что такое модель взаимодействия открытых систем, которая была разработана Международной Организацией по Стандартам (ISO, International Organization for Standardization). В соответствии с этой моделью сетевое взаимодействие делится на семь уровней.

1. Физический уровень — передача битов по физическим каналам (витая пара, коаксиальный или оптоволоконный кабель). Здесь определяются характеристики физических сред и параметры электрических сигналов.

2. Канальный уровень — передача кадра данных между любыми узлами в сетях типовой архитектуры или соседними узлами произвольной топологии. В качестве адресов на канальном уровне используются MAC-адреса.
3. Сетевой уровень — доставка (без каких-либо гарантий) пакета любому узлу в сетях произвольной топологии.
4. Транспортный уровень — доставка пакета любому узлу с произвольной архитектурой сети и заданной степенью надежности доставки. На этом уровне имеются средства для установления соединения, буферизации, нумерации и упорядочивания пакетов.
5. Сеансовый уровень — управление диалогом между узлами. Обеспечена возможность фиксации активной на данный момент стороны.
6. Уровень представления — здесь возможно преобразование данных (шифрация, компрессия).
7. Прикладной уровень — набор сетевых сервисов (FTP, E-mail и др.) для пользователя и приложения.

На рис. 14.1 вы можете увидеть, как должна выглядеть классическая сетевая модель.

Если вы внимательно прочитали предложенный список, то наверно заметили, что первые три уровня обеспечиваются оборудованием, таким как сетевые карты, маршрутизаторы, концентраторы, мосты и др. Последние три реализуются на уровне операционной системы или приложения. Четвертый уровень является промежуточным.

Как работает протокол по этой модели? Пакет попадает на прикладной уровень, и к нему добавляется заголовок. Затем пакет отправляется на уровень представления. Здесь ему также добавляется свой собственный заголовок, и пакет пересылается дальше. Так до физического уровня, который занимается непосредственно передачей данных.

Другая машина, получив пакет, начинает обратный отсчет. Пакет с физического уровня попадает на канальный, где убирается соответствующий заголовок, затем пакет поднимает выше. Уровень сети отсекает свой заголовок и передает пакет выше, и так до уровня приложения, где остается чистый пакет без служебной информации, которая была прикреплена на исходной машине перед отправкой.

Передача данных не обязательно должна начинаться с седьмого уровня. Если используемый протокол работает на 4 уровне, то процесс трансляции начнется с него, и пакет будет подниматься вверх до физического уровня для отправки. Количество уровней в протоколе определяет его потребности и возможности при передаче данных.

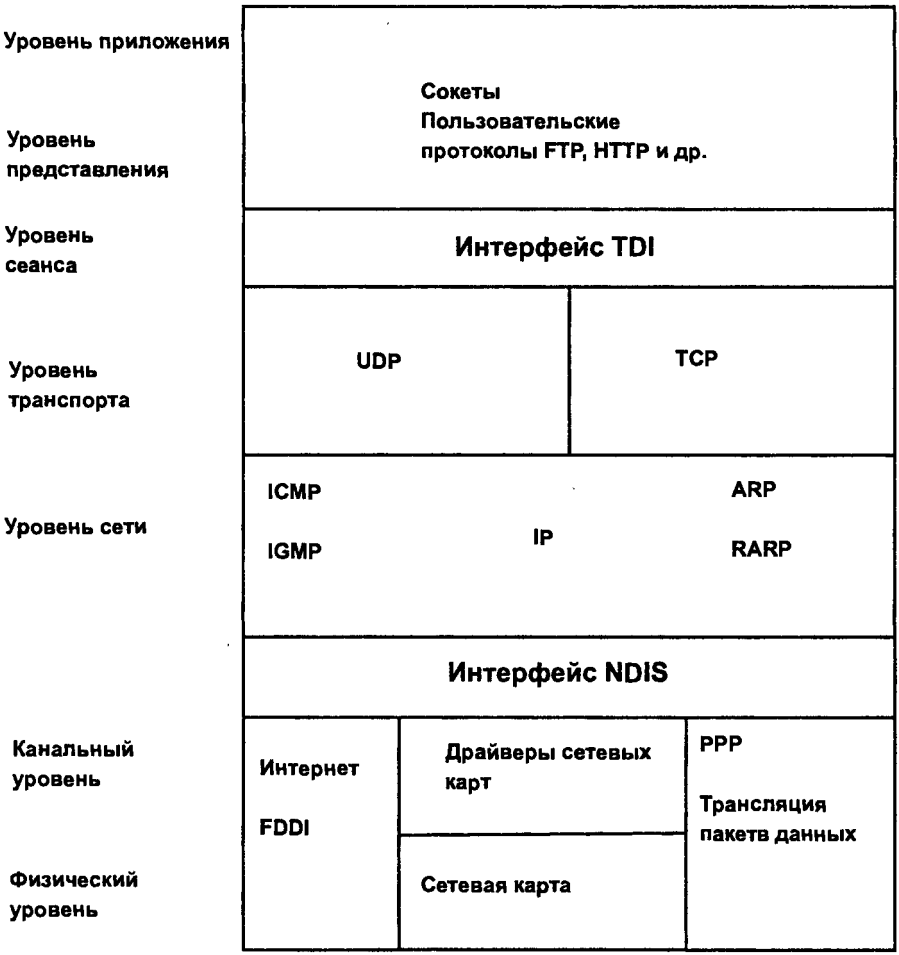

Рис. 14.1. Сетевая модель OSI

Чем ниже расположен протокол (ближе к прикладному уровню), тем шире его возможности и, соответственно, выше накладные расходы при передаче данных (больше и сложнее заголовок). Рассматриваемые сегодня протоколы будут находиться на разных уровнях, поэтому будут иметь разные возможности.

14.5.2. Пассивное подслушивание

Пассивный снифинг — прослушивание пакетов, которые проходят непосредственно через вашу сетевую карту. Такой метод удобен только при соединении компьютеров через общую шину и в сетях с топологией "Звезда", где центром выступает хаб *(см. разд. 5.2)*.

Использование пассивного сниффинга — самое простое занятие. Все пакеты, которые проходят мимо вашей сетевой карты, проверяются на принадлежность узлу. Сетевая карта сверяет адрес получателя в заголовке пакета со своим MAC-адресом, и если они совпадают, то пакет передается операционной системе для анализа. ОС читает из заголовка порт, на который направлен пакет, и далее определяет программу, открывшую порт для приема информации.

Обработка пакетов идет на уровне сетевой карты. Но эта карта может быть переведена в специальный режим, при котором все пакеты передаются операционной системе, и вы можете их увидеть с помощью специализированных программ. Необходимо заметить, что не все сетевые карты переводятся в этот режим, но, по крайней мере, современные поддерживают такую возможность.

В Интернете и в сетях, где используется Swich, такой трюк не проходит. Сетевая карта видит только свой трафик, т. к. чужие пакеты исключаются на коммутаторах или маршрутизаторах, которые установлены у провайдера. В этом случае на помощь приходит активный сниффинг.

На первый взгляд, прослушивание трафика абсолютно безопасно для хакера, и некоторые начинающие запускают программы сниффинга и сидят с ними целый день в ожидании заветных паролей. Но администратор может и должен определить наличие в сети прослушивания, даже пассивного, и наказать любознательного умельца, пока он не натворил бед.

Начнем с поиска пассивного сниффера. Для его обнаружения необходимо разослать всем компьютерам эхо-запросы (пакеты ping), в которых будет указан правильный IP-адрес, но неверный MAC-адрес. В нормальном режиме сетевая карта проверяет физический адрес, поэтому, увидев неправильное значение, пакет будет отброшен. Если на компьютере сетевая карта переведена в режим просмотра всего трафика, то пакет передается ОС, где сравнивается уже IP-адрес. Так как адрес верный, ОС откликнется. Если эхо-ответ получен, то это явно говорит о том, что на компьютере сетевая карта находится в подозрительном режиме.

Но хакеры не так уж глупы и защищаются сетевыми экранами. Достаточно запретить исходящий ICMP-трафик, и компьютер злоумышленника уже не ответит на ваши запросы, а значит, ничего определить нельзя.

Если на вашем сервере резко повысилась средняя загрузка процессора, то причиной может быть подброшенный сниффер, т. к. в этом случае сетевая карта начинает отдавать операционной системе весь проходящий трафик.

Чтобы узнать, в каком режиме работает ваш сетевой интерфейс, необходимо выполнить команду:

```
ifconfig -a
```

Если установлен параметр PROMISC, то сетевая карта работает в "неразборчивом" режиме и может прослушивать трафик.

14.5.3. Активное подслушивание

Активные снифферы делают все возможное, чтобы перенаправить чужой трафик на себя. Это достигается с помощью модификации таблиц маршрутизации и обмана сетевого оборудования.

С точки зрения реализации активный сниффер сложнее. Чтобы понять, как он работает, необходимо разобраться, как проходят пакеты на самых низших уровнях. Для передачи пакета в Интернете сетевые устройства оперируют IP-адресами. Когда вы обращаетесь по IP-адресу к какому-либо компьютеру в рамках одного и того же сегмента, используется MAC-адрес, а если адресат находится в другой сети, то пакет направляется на шлюз по умолчанию, который является маршрутизатором или компьютером, перенаправляющим пакеты на другой маршрутизатор, и он уже по IP-адресу найдет нужную сеть. Когда сеть обнаружена, внутри нее пакет следует до получателя по MAC-адресу.

Обман MAC-адреса

Как видите, внутри сети пакеты передаются только по MAC-адресу. Вот тут нам для понимания пригодится модель OSI, которую мы рассматривали в *разд. 4.5.1*. Сетевая карта, хаб и большинство коммутаторов работают на канальном уровне и оперируют только MAC-адресами. Заголовки других уровней недоступны и непонятны этим устройствам. Маршрутизаторы и коммутаторы разбирают пакет до уровня сети, где есть IP-адрес, т. е. внутри сети без этих устройств пакеты могут передаваться только по MAC-адресу.

Что же получается? В локальной сети, даже если вы отправляете данные на IP-адрес, используется физический адрес компьютера. При работе через Интернет всегда используется IP-адресация. Но просто направить в сеть пакет с IP-адресом нельзя, ведь сетевые карты не работают с IP, а пользователи никогда не указывают MAC.

Как же тогда узнать MAC-адрес получателя? Сначала отправитель пытается выяснить, у какого компьютера установлен IP-адрес в виде XXX.XXX.XXX.XXX. Для этого используется протокол ARP (Address Resolution Protocol, протокол разрешения адресов), который рассылает широковещательный запрос всем компьютерам сети и выясняет, где находится экземпляр с указанным IP-адресом. В этом пакете заполнен только IP-адрес, а вместо искомого MAC-адреса указано значение FFFFFFFFFFFFh, которое сетевая карта должна обработать и обязательно отдать на разборку операци-

онной системе. Вот тут ОС рассматривает пакет на третьем уровне, где работает ARP-протокол, и если в сети есть компьютер с запрошенным IP, то в ответном пакете будет указан его MAC-адрес. Вот теперь мы получили искомый адрес.

А кто мешает нашему компьютеру отвечать на чужие ARP-запросы и представляться участникам сети от чужого лица? Вот и я говорю, что никто. У протокола ARP нет никакой авторизации и проверки достоверности ответа. Значит пакет получит тот компьютер, который откликнется, вне зависимости от его IP.

Но неприятности еще впереди. Получив ответ на ARP-запрос, ОС кэширует результат и при последующих обращениях к IP-адресу не отправляет широковещательный запрос, а использует информацию из ARP-кэша. И вот теперь о самом страшном. Некоторые ОС (не будем указывать пальцем) кэшируют ARP-ответы не только на свои, но и на чужие запросы. Таким образом, злоумышленник может разослать информацию (ARP-ответы) со своим MAC-адресом, но чужими IP, всем компьютерам, и они добавят в кэш неверные записи.

Для просмотра текущего кэша вашей ARP-таблицы можно выполнить команду `arp`. На экране вы увидите примерно следующий результат:

```
Address      HWtype  HWaddress      Flags Mask  Iface
192.168.77.10 ether  00:03:0D:06:A4:6C C      eth0
```

Наиболее интересными колонками здесь являются:

- `Address` — IP-адрес компьютера;
- `HWtype` — тип удаленного устройства;
- `HWaddress` — MAC-адрес удаленного устройства;
- `Iface` — сетевой интерфейс.

Если вам необходимо обратиться к компьютеру с IP-адресом 192.168.77.10, то по этой таблице ОС определяет, что он находится на сетевом интерфейсе eth0, и его аппаратный адрес равен 00:03:0D:06:A4:6C.

Если вы явно видите, что адрес поддельный, то следует избавиться от неверных записей. После этого по MAC-адресу можно найти компьютер злоумышленника.

Для удаления записи необходимо использовать команду `arp` с ключом `-d` и IP-адресом. Например, для уничтожения ARP-записи, которая показана выше, необходимо выполнить следующую команду:

```
arp -d 192.168.77.10
```

Если теперь просмотреть кэш ARP-таблицы, то в результате вы увидите вместо MAC-адреса запись `(incomplete)`:

```
Address      HWtype  HWaddress     Flags Mask  Iface
192.168.77.10       (incomplete)              eth0
```

При работе с кэшем ARP вы должны знать, что записи в нем не вечны. Все, что добавляется в кэш через ARP-протокол, а не вручную, имеет статус Dynamic (динамический). Такие записи через определенное время удаляются. Хакеры знают это, поэтому могут рассылать ARP-ответы с фальшивым MAC-адресом через определенные промежутки времени. Если вы будете просто удалять каждый раз неверные записи из кэша, то эффекта это не даст. Необходимо искать злоумышленника.

Чтобы вам проще было определить поддельные записи, можно использовать протокол RARP (Reverse ARP, обратный ARP), который по известному MAC-адресу запрашивает IP-адрес. В результате вам должны вернуться ответы со всех компьютеров, IP-адреса которых находятся в вашем кэше. Помните, что записей может быть несколько. Например, в моем компьютере на сетевой карте установлено сразу два адреса для одновременной работы в двух логических сетях. Это нормальная ситуация. А вот если ответ от какого-либо IP не получен, то такую запись следует удалить.

Для управления ARP-таблицами в Windows я рекомендую использовать утилиту CyD NET Utils (**www.cydsoft.com**).

В ОС Windows проводить атаку по подделке MAC-адреса очень сложно. Если вы разошлете ложные ARP-ответы для IP-адреса 192.168.77.10, указав собственный MAC-адрес, то этот компьютер выдаст ошибку с сообщением, что адрес уже используется другим сетевым устройством, и выйдет из сети. Чтобы этого не произошло, не рассылайте широковещательные ARP-пакеты с поддельным адресом. Необходимо целенаправленно обманывать только определенную машину.

Рассылка ARP-пакетов — непростое занятие. Намного легче изменить аппаратный адрес, если это поддерживается вашей сетевой картой. Для этого используется уже знакомая нам утилита ifconfig с ключом `hw`, после чего нужно указать тип адреса и его новое значение.

С помощью подделки ARP-таблиц легко провести коммутаторы, но не маршрутизаторы, которые работают на 3 уровне, т. е. на уровне IP-адресов. Тут уже нужна фальсификация не аппаратных адресов, а IP-маршрутов. Чтобы сделать это, хакеры взламывают маршрутизаторы и перепрограммируют их на свой лад.

Противостоять подмене ARP-таблиц можно, только если вы используете для организации сети управляемые коммутаторы. В них есть возможность закре-

пить за каждым портом определенный MAC-адрес компьютера. Это усложнит задачу, но не разрешит ее полностью.

Полное решение проблемы кроется в использовании статичных ARP-записей, которые вы должны вручную прописать на каждом клиентском компьютере. Но в этом варианте кроется неудобство, потому что придется редактировать эти записи после каждого изменения сетевого оборудования на любом из компьютеров. Если где-то поменялась сетевая карта, то и ее MAC-адрес тоже изменится. Для облегчения этой процедуры можно создать сценарий на сервере, который будет заполнять ARP-таблицу нужными значениями. Клиенты должны при старте системы запускать этот сценарий.

Вывод из строя коммутаторов

Как мы уже говорили в *разд. 5.2*, концентраторы (Hub) — это устройства, которые все пришедшие с одного компьютера пакеты тиражируют на все порты, подключенные к этому хабу. Коммутаторы — интеллектуальные устройства, которые по MAC-адресу маршрутизируют трафик, проходящий через порты. Это значит, что пакеты будет видеть только получатель.

Если ваша сеть построена на основе коммутаторов, то прослушивание трафика становится невозможным. Но у Swich есть одна интересная особенность. Если он не справляется с анализом пакетов, то переключается в режим работы простого концентратора, когда входящие пакеты просто копируются всем подключенным компьютерам.

Задача хакера — загрузить коммутатор так, чтобы тот не успевал справляться с трафиком. Лучше всего это сделать, если засыпать коммутатор пакетами с неверными MAC-адресами. На анализ таких пакетов уходит слишком много ресурсов, и Swich перестает справляться с нагрузкой.

Проблема решается только установкой более мощного оборудования. В настоящее время коммутаторы таких производителей, как 3Com и Cisco, обладают достаточно мощными вычислительными ресурсами и способны преодолеть даже пиковую нагрузку ложных пакетов. Оборудование других производителей мне тестировать не приходилось.

Обман маршрутизатора

Маршрутизаторы тоже можно обманывать, точнее сказать, можно провести компьютер. Допустим, что ваша сеть разбита на несколько подсетей и использует несколько маршрутизаторов. На рис. 14.2 показан пример такой сети.

Если компьютер из сети 1 отправляет пакет другому компьютеру своей сети, то, как мы знаем, посылка осуществляется по MAC-адресу, и маршрутизатор

не используется. Если пакет предназначен другой сети, то он направляется на шлюз по умолчанию. Допустим, что в качестве шлюза выступает сетевой экран. В этом случае, если компьютер 1 адресует пакет в Интернет, то Firewall просто перенаправляет его в сеть. А что если получателем пакета выступает компьютер из сети 2? Будет ли сетевой экран пересылать его маршрутизатору? Не обязательно. Сетевой экран в таком случае может вернуть ICMP-сообщение с предложением компьютеру самостоятельно работать с маршрутизатором, соединяющим сети 1 и 2.

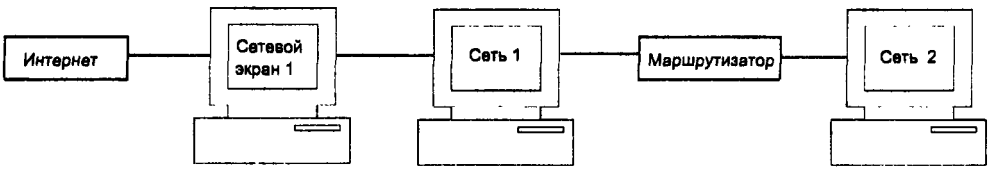

Рис. 14.2. Сеть с двумя маршрутизаторами

Так как ICMP-протокол не использует авторизацию и шифрование, злоумышленник самостоятельно может направить такой пакет любому клиенту и попросить его работать не с маршрутизатором, а с его компьютером. В результате пакеты пойдут через компьютер хакера, и он сможет их просмотреть.

Я рекомендую отключить перенаправление маршрутизации. Для этого установите `1` в файле **/proc/sys/net/ipv4/conf/all/accept_redirects,** выполнив команду:

```
echo 1 > /proc/sys/net/ipv4/conf/all/accept_redirects
```

Можно изменить этот параметр и через файл **/etc/sysctrl.conf**, добавив в нем строку:

```
nt.ipv4.conf.all.accept_redirection=0
```

Если в вашей сети только один маршрутизатор, то ничего кроме повышения безопасности не произойдет. Даже если у вас два или более таких устройств, на работе сети это сильно не скажется. Разве что трафик будет идти через два маршрутизатора, а не напрямую.

Так как для проведения атаки необходим ICMP-протокол, то его можно запретить с помощью сетевого экрана, и хакер не сможет направить вам сообщение о перенаправлении маршрутизатора.

14.5.4. Перехват соединения

Атака на компьютеры с помощью перехвата соединения была первый раз применена еще несколько десятков лет назад, и до сих пор единственным

эффективным методом борьбы с ней является использование шифрования пакетов.

Когда компьютеры по протоколу TCP устанавливают связь друг с другом, на начальном этапе вводятся два счетчика, увеличивающихся с каждым отправленным пакетом. Это позволяет проверять целостность соединения. С помощью сниффера этот счетчик может быть подслушан, что в определенный момент позволит хакеру перехватить соединение и стать его владельцем, общаясь с сервером вместо клиента. Компьютер, который устанавливал соединение, в этот момент теряет связь с сервером.

Опасность атаки заключается в том, что хакер таким образом обходит все механизмы аутентификации. Легальный клиент авторизуется на сервере, а хакер после этого перехватывает соединение на себя и использует его в своих целях.

Проблема незащищенности соединения кроется в устарелости протокола TCP/IP. Его создатели не ожидали, что сеть будут прослушивать, и злоумышленники смогут обойти простую защиту соединений, встроенную в протокол. Вопрос решится с переходом на протокол IPv6.

14.5.5. Защита от прослушивания

Несмотря на то, что можно определить, что вас прослушивают, иногда это может оказаться слишком поздно. Пока вы ищете хакера, он может успеть поймать пакет с паролями и взломать систему. Если прослушиванием занимается программа, которая установлена на взломанный компьютер, то можно вычислить только этот компьютер и его владельца, но не факт, что вы найдете самого хакера.

Получается, что бороться с прослушиванием такими методами нельзя. Необходимо сделать так, чтобы этот способ не дал результатов, и хакер перестал даже думать об этом. Полноценное решение проблемы таится в шифровании любого трафика.

Доверяться сети и передавать открытые данные уже нельзя. Прошли те времена, когда сети были только для профессионалов, использующих их исключительно по назначению. В наше время в Интернете можно встретить кого угодно, от ребенка до пенсионера, от школьника до ученого. В сети сейчас работают не только добропорядочные граждане, но и молодые любопытные ребята, и даже преступники.

В *разд. 5.2* мы рассмотрели, как можно шифровать трафик любого сервиса. Вы можете закодировать любые соединения и должны это делать, если передается секретная информация.

Но прежде, чем организовывать зашифрованные каналы, необходимо убедиться, что этого не сделали до вас. Нет, я не говорю, что кто-то зашифровал ваш трафик по FTP-протоколу, я имею в виду готовые технологии. Так, для протокола HTTP уже давно существует реализация с поддержкой шифрования — HTTPS. Вы можете использовать протокол HTTP для передачи открытых данных, которые итак доступны для всеобщего просмотра (публичные Web-страницы), а секретную информацию (номера кредитных карт) можно передавать через HTTPS.

Если хакер перехватит зашифрованный пакет, то не сможет его расшифровать без закрытых ключей, которые получить практически невозможно. Единственный способ раскрыть данные — подбор пароля, но это отнимет слишком много времени, и не факт, что перехваченный пакет содержит необходимую злоумышленнику информацию.

14.5.6. Практика прослушивания

Прочитав этот раздел, складывается впечатление, что прослушивание сети приносит только беды и является оружием хакеров. Может даже показаться, что производители оборудования должны сделать аппаратную защиту от прослушивания.

Это не совсем так. Программы снифферы разрабатывались для программистов и администраторов и являются очень удобным средством для отладки работы различных протоколов.

Я не буду рассматривать все известные мне программы прослушивания трафика, потому что их много, и каждая из них обладает своими особенностями. Но хочу обратить ваше внимание на мою любимую (и одну из самых мощных) — hunt. С ее помощью можно подслушивать трафик, подменять ARP-записи и даже перехватывать соединения.

Очень популярным, а в некоторых случаях и более удобным, является пакет dsniff. Он включает свыше десяти утилит и может использоваться для решения любых, как администраторских, так и хакерских задач. Подробней о составе этого пакета можно узнать из *приложения 2*.

14.6. DoS/DdoS-атаки

Одной из самых страшных является DoS-атака. На мой взгляд, это самое глупое, что могли придумать хакеры. Когда не получается взломать сервер, его делают недоступным различными методами, вплоть до бессмысленного замусоривания канала связи. В этом разделе мы и рассмотрим основные из этих способов.

Существует разновидность DoS-атаки — DDoS (Distributed DoS, распределенная DoS), отличается она тем, что штурм производят сразу множество компьютеров.

Самое страшное заключается в том, что от данного типа атаки иногда невозможно защититься, особенно от DDoS. Если на сервер поступает слишком много запросов, то он не успевает их обрабатывать, и не все смогут получить доступ к серверу. Представьте себе, если бы все компьютеры планеты одновременно попытались обратиться даже к самому могучему серверу (кластеру из серверов). Ни один канал связи не сможет пропустить такого количества соединений, даже у таких мощных серверов, как **yahoo.com google.com**, не хватит мощностей. Таким образом, далеко не все пользователи смогут работать с сайтами.

Давайте разберем основные атаки DoS и DDoS и посмотрим, с какими можно бороться, а с какими — бесполезно. К тому же, нас будут интересовать методы притивостояния этим атакам.

14.6.1. Ping of Dead

Мы уже знаем, что с помощью утилиты ping можно проверить соединение с удаленной системой *(см. разд. 5.1)*. Для этого используется ICMP-протокол. Когда сервер получает ICMP-сообщение с кодом Echo Request, то он должен ответить таким же сообщением, с тем же количеством данных.

В некоторых ОС обнаружились ошибки обработки ping-пакетов. Просто программисты, которые реализовывали ICMP-протокол, подразумевали, что пользователи будут применять его правильно и не станут посылать слишком большие пакеты. Надежда на добросовестность пользователей обернулась появлением атаки Ping of Dead (Ping смерти). Хакеры формировали такие пакеты, которые сервер обрабатывал неверно и зависал. Наиболее нашумевшая атака вылилась в посылку пакетов размером более 64 Кбайт. Программисты зарезервировали под принимаемые данные только 64 Кбайта, и этого не хватило для приема пакетов. На сервере это может выглядеть как переполнение буфера.

Борьба с такими атаками может быть реализована только средствами сетевого экрана. Можно запретить получение ICMP-пакетов с кодом Echo Request и атака Ping of Dead на вашу систему не принесет результата. А затем обновите ОС.

14.6.2. ICMP flood

Когда не получается уничтожить сервер интеллектуальными способами, взломщики начинают использовать flood — засыпание сервера ICMP-паке-

тами. Вот это самый идиотский вид атаки, но легко загружает канал связи. Для его реализации достаточно половины канала штурмуемой системы.

Допустим, что у сервера есть канал связи с пропускной способностью 64 Кбит/с. Вам достаточно половины, чтобы полностью его загрузить. Для этого непрерывно посылаем на сервер ping-запросы с большим размером пакета. Желательно, чтобы в качестве отправителя стоял не ваш адрес, а любой другой. Если вы своими ping-пакетами создадите нагрузку на канал сервера в 32 Кбита/с, то вторая его половина будет занята ответами на несуществующий или чужой адрес.

Защита от этой атаки такая же, как и от Ping of Dead, а именно запрет ICMP-трафика. Благо этот протокол не очень нужен, особенно входящий из Интернета ICMP-трафик.

14.6.3. TCP SYN

У большинства серверов есть ограничение на количество подключений. В некоторых случаях это связано с используемой технологией, но может быть и лимит, определенный в настройках конкретного сервера.

Из названия атаки можно догадаться, что задача хакера — направить на сервер большое количество TCP-пакетов с установленным флагом SYN. Эти пакеты используются для установки подключения к серверу. Таким образом, достигнув ограничения, сервер больше не будет принимать новые подключения со стороны клиентов.

Спрятаться от подобной атаки собственными силами практически невозможно. Вам остается только с помощью сетевого экрана ограничить SYN-пакеты, но от этого защита не станет лучше.

Как временное решение, можно увеличить лимит подключений со стороны клиентов. Нагрузка на сервер от этого не возрастет, ведь хакер только инициализирует свои соединения, но не посылает какие-либо пакеты. Но ограничение в конфигурационном файле существует не всегда. Это может быть следствием используемой программой технологии.

Следующий способ защиты заключается в уменьшении времени действия неактивного соединения. В некоторых программах есть возможность в конфигурационных файлах установить длительность неактивного соединения. Убавив это значение до 10 секунд, сервер не получится загрузить SYN-пакетами. Пока образуется очередь подключений, первые соединения уже будут разрываться. Конечно же, могут возникнуть проблемы и неудобства у добропорядочных пользователей, которым, возможно, придется лишний раз устанавливать соединение с сервером, но, по крайней мере, работа не остановится.

Самая лучшая защита должна быть выполнена программистом. Как минимум, он должен реализовывать в своих программах две возможности: настройку ограничения на количество подключений и разрыв неактивных соединений через указанное в конфигурации время. Необходим запрет на установку нескольких подключений с одного и того же IP-адреса.

14.6.4. TCP flood

Эта атака аналогична ICMP flood. Если не помогает смекалка, и хакер не может найти уязвимого места, то начинается засыпание сервера бессмысленными TCP-пакетами. Их эффективность иногда даже ниже ICMP-пакетов. Если в случае использования ping-запросов сервер обязан вернуть пакет с такими же данными, то в TCP-протоколе это не обязательно. Таким образом, канал связи у хакера должен быть идентичным, а то и больше, чем у атакуемой системы.

В случае использования HHTP-протокола можно загрузить сервер даже с маленького канала. Необходимо посылать такие запросы, которые требуют на сервере большой нагрузки. Например, если необходимо вывести из строя поисковую систему, можно направить на сервер множество запросов на поиск особо популярных слов. Если серверные сценарии реализованы неэффективно, то обработка такого задания может занять слишком много времени.

Можно использовать скачивание большого файла по протоколу HTTP. Множество таких запросов при плохой организации кэширования на сервере также могут привести к зависанию.

Но у TCP-протокола есть преимущество. В большинстве сетей внешний ICMP-трафик нейтрализуется сетевыми экранами, а вот TCP-пакеты обязательно остаются открытыми для общедоступных ресурсов, и их блокировать невозможно.

Конечно же, организовать успешную атаку на мощный сервер в одиночку нереально, а вот большим количеством компьютеров можно добиться серьезных результатов.

14.6.5. UDP

Ошибки в программах, ориентированных на UDP, наиболее опасны, потому что этот протокол не использует виртуального соединения. По нему пакеты просто отправляются по сети, и нет никаких механизмов проверки достоверности указанных данных. Если при использовании TCP подделать IP-адрес достаточно сложно, то с помощью UDP-протокола это реализуется элементарно.

Благо UDP-протокол на публичных серверах используется редко, есть возможность его запретить с помощью сетевого экрана. Если протокол необходим, то защиту может построить только программист, создав на основе UDP хотя бы какую-то проверку подлинности получаемых пакетов.

14.6.6. DDoS

На данный момент можно сказать, что именно за DDoS будущее. Ошибки в программах, которые позволят несколькими пакетами убить сервер, с каждым днем уменьшаются и появляются все реже, потому что программисты теперь больше внимания уделяют безопасности при написании сетевых программ. А вот от DDoS никуда не денешься, и от этой атаки реальной, универсальной и эффективной защиты нет и сложно себе представить.

С другой стороны, реализовать действительно массовую атаку DDoS очень сложно, и крупные компании были даже уверены, что это практически невозможно. Даже очень большая группа хакеров на самых быстрых каналах не сможет получить в свое распоряжение такие вычислительные ресурсы, как у серверов **yahoo.com**, **google.com** или **microsoft.com**. Но хакеры находят новые способы. Мало кто из пользователей добровольно отдаст мощность своего компьютера для проведения распределенной атаки на крупные серверы. Чтобы решить эту проблему, хакеры пишут вирусы, которые без разрешения занимаются захватом.

Ярким примером удачной распределенной атаки является червь MyDoom. Этот червь, начиная с 22 августа 2003 года, в течение трех дней атаковал сайт компании SCO со всех зараженных в Интернете компьютеров. Через некоторое время подобную атаку пытались реализовать через вирус MyDoom B, но на серверы компании Microsoft. Вторая атака оказалась менее эффективной, но я это связываю с тем, что меньшее количество компьютеров в Интернете было заражено этим вирусом, поэтому мощности не хватило для проведения полноценного налета, да и код червя был далек от идеала.

Нередко для реализации DDoS хакеры используют захваченные мощные серверы на высокоскоростных каналах связи. Это позволяет злоумышленнику получить необходимые ресурсы для удачной распределенной атаки.

В будущем можно ожидать новые атаки DDoS, которые окажутся более эффективными и оригинальными. Здесь уже администратор будет бессилен, и в данном случае на помощь должны приходить правоохранительные органы.

14.6.7. DoS

Если вы уже посещаете бюллетени BugTraq, то должны заметить, что уязвимости, дающие повод для проведения атаки DoS, регулярно появляются в

различных сетевых программах. К этой категории очень часто относят любые ошибки переполнения буфера, с помощью которых можно сделать сервер недоступным.

О переполнении буфера мы уже говорили в *разд. 14.2* и знаем, что с этими ошибками можно бороться, даже не дожидаясь исправлений программистами. Достаточно только поставить патч на ядро ОС, при котором запрещено выполнять код из стека.

Самое странное, что количество ошибок не уменьшается, они идентичные во многих программах, а иногда совершаются одними и теми же людьми. В настоящее время существует множество книг и документов, в которых описаны все проблемы, связанные с переполнением буфера. И несмотря на это программисты продолжают совершать ошибки. Это говорит только о низком качестве образования.

Мне кажется, что качество программ падает из-за широкого распространения офшорного программирования, особенно если используются специалисты из малоразвитых стран, где и уровень подготовки низкий, и работники готовы получать мизерную зарплату, и, как следствие этого, — невозможность создать действительно качественный код. Из-за этого администраторам приходится сталкиваться с одними и теми же задачами.

Проблема может решиться, когда разработчики программного обеспечения начнут использовать более квалифицированные людские ресурсы.

ОС Linux поставляется в исходных кодах, и абсолютно любой доморощенный программист может вносить изменения. Когда она создавалась множеством энтузиастов, то ошибок было очень много, потому что отсутствовал четкий контроль.

В настоящее время уже мало дистрибутивов, выпускающихся стихийными группами программистов. Все, кто захотели делать качественные продукты, организовали фирмы. Теперь уже изменения, внесенные программистом-одиночкой, попадут в дистрибутив, только если его разработчик решит, что код безопасен и полезен для пользователей. Благодаря этому надежность Linux повышается, чего нельзя сказать об отдельных ее компонентах.

14.6.8. Защита от DoS/DDoS

Самая эффективная защита от атак DoS, проводимых через ошибки в программах, — это своевременное обновление этих программ. А вот если хакер направил свои усилия на полное поглощение ресурсов сервера, то в данном случае защититься сложно, но необходимо сделать все возможное, чтобы усложнить задачу злоумышленнику.

Сначала необходимо определить самое слабое место сервера. Для этого можно искусственно заставить его работать на максимальной нагрузке. Это можно сделать с помощью большого количества пользователей, которые будут обращаться к серверу, или написать специализированную программу, эмулирующую эту ситуацию.

Во время пиковой нагрузки необходимо проследить, каких ресурсов не хватает компьютеру. Обратите внимание на следующие компоненты:

- пропускную способность сети;
- пропускную способность сетевого оборудования;
- загрузку процессора;
- нагрузку на жесткий диск;
- загрузку оперативной памяти.

Определите самые узкие места вашей системы, чтобы увеличить производительность. Нет смысла наращивать внешние каналы связи до 100 Мбит/с, если ваша внутренняя сеть работает только на скорости 10 Мбит/с. Это излишняя трата денег, а проблема останется. Хакеру достаточно будет направить на сервер любой трафик, чтобы забить 10 Мбитный канал мусором, и ресурсов вашей внутренней сети не хватит для обработки такого объема данных. Именно поэтому очень важно правильно определить слабое звено в вашей системе.

Настройте сетевые интерфейсы и параметры ОС на максимальную производительность *(см. разд 14.11)*. Это значит, что не должно быть лишних затрат, особенно сетевых. Понизить расходы на обработку сетевых пакетов можно с помощью полного запрета ICMP-трафика.

14.7. Проникновение через доверительные узлы

Когда хакер хочет проникнуть на сервер в какой-либо сети, но не может подступиться к нему, то чаще всего приходится прибегать к использованию доверительных компьютеров. Все машины в сети не могут быть защищены абсолютно одинаково. Задача хакера — найти уязвимый компьютер и попробовать проникнуть на сервер через него.

Чтобы решить поставленную задачу, необходимо определить, какие компьютеры в сети сейчас работают. Для этого можно использовать классическую команду `ping`, вручную запуская ее для каждого IP-адреса целевой сети, но лучше воспользоваться утилитой nmap. Она может автоматически прозондировать (`ping sweep`) указанный диапазон IP-адресов.

Для сканирования диапазона адресов можно воспользоваться следующей командой:

```
nmap -sP 192.168.1.0/24
```

После указания IP-адреса стоит знак "/", за которым следует маска сети. Таким же образом мы задавали маску при настройке сетевого экрана. В ответ на директиву программа пошлет ping-запросы на все компьютеры сети и отобразит нам доступные в настоящее время.

Использование ping-пакетов — удобный и быстрый способ просканировать сеть, но может выдать неверный результат, если атакуемая сеть защищена сетевым экраном и в ней фильтруются ping-пакеты. Но Firewall, установленный на подступах к сети, может защитить только от сканирования извне. Для предотвращения внутреннего зондирования необходимо настраивать сетевой экран на каждом компьютере сети. Можно отключить службу, отвечающую на ping-запросы, но ничего не спасет от прощупывания портов.

После того как определены все работающие компьютеры, хакер начинает сканировать каждый из них, определяя уязвимые сервисы. Взлом множества компьютеров намного проще, потому что хотя бы один может поддаться, не выдержав натиска хакера.

После проникновения на один из компьютеров сети хакер может повторить попытку зондирования с захваченной машины. Так как этот компьютер находится внутри сети, то он может дать более точный результат, потому что это сканирование не будет проходить через сетевой экран.

При наличии контроля над одним из компьютеров сети процесс взлома упрощается, если существует вероятность следующих событий:

- компьютер находится в доверительных отношениях с сервером. В ОС Linux можно указать такие компьютеры, и с них подключение происходит без каких-либо паролей. Никогда не используйте доверительных отношений, потому что это удар по безопасности! Именно поэтому мы не затрагивали эту тему в данной книге;
- пароль доступа к взломанному компьютеру подойдет для входа на основной сервер, или в файле **/etc/passwd** прописаны пользователи, которые так же работают с сервером. Такое бывает очень часто. Пользователи не любят запоминать много паролей, поэтому используют одни и те же параметры для подключения к любому компьютеру в сети.

Конечно же, если удастся получить контроль над учетными записями, то это еще не гарантирует, что хотя бы одна из них будет иметь права администратора на основном сервере. Но иногда достаточно даже минимального статуса, чтобы потом повысить права через уязвимость.

Проблема использования одного пароля для доступа к разным серверам затрагивает не только простых пользователей. Администратор может использо-

вать пароль fxdgdfg и имя root на основном сервере, а на других компьютерах этот же пароль может быть в комбинации с реальным именем администратора. Взломщики собирают все найденные пароли и потом используют их для подбора пароля пользователя root.

Если честно, то я сам грешу проблемой одного пароля для разных сервисов. Но одинаковые комбинации устанавливаю только там, где это не может причинить вред, например, когда регистрируюсь на форумах или на сайтах, собирающих какую-то статистику. А вот администраторские пароли у меня разные для каждой системы.

Вы должны защищать все компьютеры в равной мере и использовать различные пароли для пользователей с правами администратора.

14.8. Небезопасная 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.

Как я уже говорил, удобство и безопасность — несовместимые вещи, а NFS слишком удобна.

В состав NFS входит утилита showmount, которая может отобразить, какие директории и какими пользователями подключены. Для администратора это неоценимая информация.

Выполните команду `showmount -a localhost`.

Вы сможете увидеть следующую информацию о NFS на своем сервере:

```
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 символом двоеточия. В первой находится имя компьютера, подключившего удаленный раздел, а во второй — путь на сервере к подсоединенному ресурсу.

Приятно видеть подробную информацию, но и опасно, потому что команда может выполняться удаленно, а значит, любой хакер доберется с ее помощью до следующей информации:

- подключенные директории. В примере выше подсоединяются различные папки из раздела **/home**. Чаще всего их названия совпадают с именами пользователей, поэтому легко определить действительные имена юзеров, не обращаясь к файлу **/etc/passwd**. С такой информацией хакеру проще будет подбирать пароли доступа;
- имена компьютеров в сети. Если вы потратили большие усилия на защиту своего DNS-сервера, то можете считать, что вы сделали это зря, если на каком-либо сервере установлена NFS. Один запрос показывает имена компьютеров в сети, пусть и не все, а только работающие с NFS, но и этого может быть достаточно для хакера. Кстати, ему не надо даже зондировать сеть с помощью ping-запросов, потому что и так видно действующие компьютеры;
- используемые программы, включая номер версии. Если пользователи монтируют каталоги с программами, то имена этих каталогов могут выглядеть как **/usr/local/jail 1.0**. Это только пример, но он показывает, что директории в Linux могут содержать в качестве имени название программы и, самое главное, номер версии.

В зависимости от того, какие открыты каталоги, хакер может получить намного больше информации. Выходит, что утилиты NFS слишком болтливы, а этого нельзя допускать.

Если вы решили использовать NFS, то позаботьтесь о том, чтобы она не был доступна из Интернета. Для этого необходимо запретить подключение к 2049 UDP- и TCP-порту извне. Эти функции может выполнить сетевой экран. А если хакер уже взломал какой-то компьютер в сети и получил возможность выполнять команды внутри нее, то защита сетевого экрана не поможет.

При настройке NFS в файле **/etc/exports** указываются экспортируемые файловые системы и права доступа к ним. Никогда не открывайте полный доступ ко всей системе, т. е. в файле не должно быть строки:

```
/    rw
```

Необходимо четко прописывать пути к каталогам, которые могут быть монтированы пользователями. Это значит, что если пользователи должны иметь возможность подключать домашние каталоги, то следующее разрешение также является неверным и опасным:

```
/home    rw
```

В чем здесь опасность? Не все пользовательские каталоги должны монтироваться удаленно. Например, если вы работаете под пользовательской учетной

записью, но являетесь администратором, то в вашем каталоге могут быть программы, используемые для управления системой. Нельзя допустить, чтобы злоумышленник смог его увидеть (даже с правами только на чтение). Разрешайте подключение только 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 постановкой 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. гл. 6)*. Этот сервис менее болтлив и может решить ваши потребности в разделении каталогов сервера.

14.9. 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 всегда и с любой системой, но вы должны уметь их раскрывать.

Как можно выявить хакера? Существует очень много методов, и сейчас мы рассмотрим наиболее интересные и эффективные.

14.9.1. Осведомлен, значит защищен

Очень часто я использую 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. 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.

Получив сообщение о том, что была выполнена опасная команда, и она запускалась не администратором, есть 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 узнать, есть ли там администратор. Определить место расположения программы можно командой:

```
which who
```

В результате вы должны увидеть путь типа **/usr/bin/who**.

Для начала запоминаем права на файл, выполнив команду:

```
ls -al /usr/bin/who
```

Для данной программы должны быть права -rwxr-xr-x, что соответствует числу 755.

Теперь необходимо переименовать файл **/usr/bin/who** в **/usr/bin/system_who**. Это можно сделать следующей командой:

```
mv /usr/bin/who /usr/bin/system_who
```

Меняем права доступа:

```
chmod 755 /usr/bin/system_who
```

Теперь, чтобы выполнить команду who, нужно использовать имя system_who. Но скопированный файл может стать неисполняемым, поэтому второй командой мы восстанавливаем права.

Затем создаем заглушку для программы who. Это будет файл с именем who, в директории **/usr/bin**. Когда хакер будет выполнять команду who, то будет запускаться наш файл. Для этого выполним команду:

```
cat /usr/bin/who
```

Теперь все команды, вводимые с консоли, будут записываться в файл **/usr/bin/who**. Наберите две строки:

```
/usr/bin/system_who
id | mail -n -s attack root@FlenovM
```

После этого нажмите сочетание клавиш <Ctrl>+<D> и измените права на созданный нами файл **/usr/bin/who**, установив значение 755.

Выполните команду who. Все вроде нормально, но если проверить почту, то в вашем ящике будет лежать новое письмо с заголовком "attack" (рис. 14.3), и в нем будут находиться параметры (все, что вернет команда id) пользователя,

выполнившего команду. Это из-за того, что запустилась не системная команда, а наш файл, который содержит две строки:

- `/usr/bin/system_who` — сначала запускаем системный файл who, который мы переименовали, чтобы взломщик ничего не заподозрил;
- `id | mail -n -s attack root@FlenovM` — выполняется команда `id`, а результат направляется с помощью почтовой программы mail в почтовый ящик root@FlenovM. Ключ `-s` задает заголовок письма. Ключ `-n` предотвращает чтение файла **/etc/mail.rc**. Я рекомендую указывать только эти атрибуты, чтобы на экране не появлялось лишней информации, и взломщик ничего не заподозрил. Хакер не должен знать, что программа отправила администратору какое-нибудь сообщение.

Таким образом, можно подменить все опасные программы, которые должны быть недоступны простым пользователям.

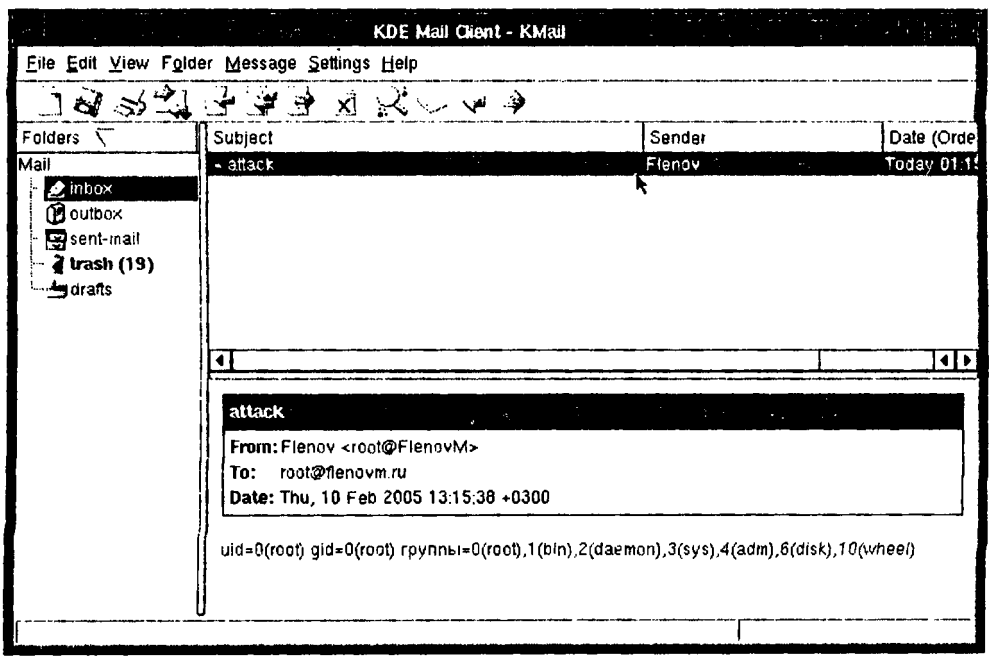

Рис. 14.3. Пример сообщения об атаке

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. Программы, написанные на языке C и откомпилированные в ис-

полняемый файл, при просмотре обнаруживают абсолютно ни о чем не говорящий мусор.

14.9.2. Ловля на живца

В *разд. 4.11* мы затронем тему построения сети, где публичные ресурсы вынесены на самостоятельные серверы и защищаются отдельно от основной сети. Мы уже коснулись вопроса создания серверов-приманок, и сейчас настало время поговорить об этом более подробно.

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. 14.4 приведена схема классической honeypot-сети. От Интернета ее отделяет сетевой экран, за которым находится публичная сеть (общедоступные ресурсы и подставные серверы/компьютеры). Далее идет второй сетевой экран, который защищает и скрывает приватную сеть.

Рис. 14.4. Построение honeypot-сети

Как только хакер попадает в капкан, администраторы начинают идентификацию и его 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 сложных срабатываний, его защита должна быть достаточной, дабы сервер нельзя было сломать программами, автоматизирующими поиск уязвимостей. Иначе число ежедневно пойманных в ловушку хакеров будет исчисляться сотнями, а то и более, ведь количество сканирований популярного ресурса велико.

Я настраиваю свои honeypot-серверы на максимальную безопасность, просто сетевой экран, который их защищает (на рис. 14.3 это Firewall 1), пропускает практически любой трафик.

Это позволяет решить сразу несколько потенциальных проблем:

- не вызывает подозрений у хакера. Слишком открытые серверы с дырявыми сервисами насторожат профессионала, и он не будет трогать такие компьютеры;
- уменьшается количество срабатываний системы на каждого начинающего хакера, который случайно нашел программу для взлома;
- позволяет выявлять атаки, которые еще неизвестны специалистам по безопасности, и защищаться даже от них. Если сервер с правильными настройками взломан, значит, компьютеры за пределами второго сетевого экрана тоже уязвимы, но вы еще не знаете об этой лазейке или могли совершить ошибку в конфигурировании.

Как только вы заметили взлом поддельного сервера, выполняются следующие шаги:

- анализируется уязвимость, через которую проник хакер. Найдя ошибку в конфигурировании или дырявый сервис, необходимо отыскать заплатку и установить ее на компьютеры защищенной сети, чтобы хакер после взлома второго сетевого экрана не смог воспользоваться этой уязвимостью;
- выясняется источник угрозы, IP-адрес хакера, и вся найденная информация сообщается в правоохранительные органы.

Так как на honeypot-компьютерах сервисы не обрабатывают, а только эмулируют реальные подключения пользователей, то для таких серверов нет необходимости устанавливать мощное оборудование. Достаточно устаревшего железа, которое вы уже давно не используете. В любой организации кабинет администратора завален старыми системными блоками, которые лежат только как запасные части.

Если у вас нет старых компьютеров, то в honeypot можно превратить и серверы, которые используются в качестве публичных. В принципе, каждый публичный сервер итак должен содержать мощные системы журналирования и мониторинга, и от honeypot их должно отличать отсутствие заведомо открытых уязвимостей и легких паролей.

Чтобы хакер ничего не заподозрил и обратил внимание на подставные компьютеры, одной защиты мало. Необходимо, чтобы honeypot создавали видимость действия и обрабатывали какой-либо трафик. Для этого могут быть написаны специальные утилиты, которые с определенным интервалом производят соединения между подставными компьютерами, имитируя работу пользователей. Компьютер, который находится в сети и ничего не делает, вызовет подозрение, и его будет ломать только "чайник".

14.10. Взлом паролей

Очередной идиотизм, который могли придумать хакеры, — это подбор паролей. Когда взломщик не может изобрести ничего оригинального, он запускает подбор паролей. Об этом мы уже говорили в *разд. 1.1.7*, но тогда мы затронули эту тему только вскользь, а сейчас поговорим более подробно. Нам предстоит свести всю отрывочную информацию, которую мы получали в предыдущих главах, в одно целое и узнать еще много нового по этому вопросу.

14.10.1. По словарю или все подряд

Существует два метода подбора паролей: по словарю и полный перебор всех возможных вариантов. Рассмотрим каждый из них.

Подбор по словарю. В этом случае хакер сначала готовит файл с наиболее часто используемыми для паролей словами. Затем запускается программа подбора паролей, которая последовательно проверяет все термины из указанного словаря.

Преимущество этого метода состоит в том, что пароль может быть найден достаточно быстро, если он простой и есть в словаре. Взломщик для подбора может использовать, например, словарь английского языка, и количество возможных вариантов не превысит 20 000 слов (я имею в виду наиболее часто используемые).

Задача хакера — составить как можно более эффективный словарь, который при этом может быть и незначительного размера. Для этого сначала собирается вся возможная информация об администраторе: имя и фамилия его и родственников, день рождения, клички домашних животных, интересы, любимые фильмы, музыка и т. д. Полученные данные добавляются в начало словаря. Все исследовательские конторы пришли к одному и тому же выводу, что большинство людей выбирают в качестве пароля имена своих любимых собачек, кошечек, знаменательные даты, номера телефонов, а чаще всего что-то связанное с хобби. Хорошо подобранный словарь может сломать практически любую систему, т. к. всегда найдутся неопытные пользователи с такими паролями.

С другой стороны, если использованы все рекомендации по подбору пароля (длинный и сложный, состоящий из букв и символов, да еще с использованием разного регистра), то вероятность того, что такая комбинация окажется в словаре хакера, стремится к нулю. И поэтому время, затраченное злоумышленником, окажется зря потерянным.

Полный перебор. В этом случае программа последовательно перебирает все возможные комбинации букв, цифр и символов в различных регистрах.

Количество вариантов при таком алгоритме исчисляется миллиардами и прямо пропорционально зависит от длины пароля.

Метод дает 100 % гарантию удачного подбора, но реализация может отнять недели или даже месяцы. Если администратор системы ежемесячно меняет пароли, то пока хакер занимается подбором, результат уже устаревает.

14.10.2. Удаленно или локально

Подбор паролей разделяют на удаленный и локальный. В первом случае хакер пытается подключиться к компьютеру по сети с разными паролями. Это наиболее опасный для него метод, потому что каждая неверная попытка регистрации записывается в журнал безопасности. Если вы хотя бы иногда заглядываете туда, то уже на раннем этапе сможете найти злоумышленника и предотвратить подбор с помощью запрета подключения с IP-адреса хакера. Есть еще одна проблема — подбор осуществляется к определенному сервису, но нет гарантии, что другая программа тоже будет использовать этот пароль. Кроме того, большинство сервисов имеют настройки, лимитирующие попытки входа. И если хакер не уложится в это ограничение, то соединение разрывается, и его приходится восстанавливать. Каждое подключение — это лишние затраты времени, которые увеличивают время подбора даже по словарю.

Для того чтобы подбор оказался долгим занятием, в некоторых сервисах есть возможность при неправильном указании параметров доступа искусственно делать задержку между попытками ввода. Ярким примером является сама ОС. Попробуйте при регистрации в Linux указать неверный пароль или имя пользователя. ОС будет производить проверку намного дольше, чем при указании правильных параметров. Чем больше задержка, тем больше времени необходимо хакеру для подбора пароля.

Но это препятствие довольно просто обойти. Достаточно запустить несколько потоков подбора, которые будут параллельно подключаться к серверу.

Наиболее эффективный метод защиты в этом случае — запрет подключения с IP-адреса к серверу с помощью сетевого экрана. Если вы заметили, что кто-то пытается подобрать пароль, немедленно определите IP-адрес злоумышленника и добавьте запрещающий фильтр в сетевой экран.

Из-за сложности подбора пароля хакеры стремятся получить доступ к файлу **/etc/shadow** и скачать его на свой компьютер, чтобы работать с ним локально. В этом случае операция происходит намного быстрее по следующим причинам:

- хакеру становятся известны реальные имена пользователей, которые зарегистрированы на сервере;

- защита сервера уже не может ничем помочь, потому что вы потеряли контроль над паролями;
- т. к. пароли хранятся в файле в зашифрованном виде, достаточно один раз зашифровать его и сравнить со всеми кэш-суммами. Чем больше в файле записей, тем больше вероятность, что хотя бы один пароль подойдет;
- в больших файлах паролей существует вероятность, что хотя бы один пользователь задал в качестве пароля регистрационное имя.

Локальный взлом намного быстрее и безопаснее, единственная проблема заключается в том, как получить файл **/etc/shadow**. Этот файл доступен для чтения и записи только администратору root, а все остальные пользователи не могут его увидеть.

14.10.3. Защита

Защиты от подбора пароля в принципе нет и не может быть. Если хакер получит доступ к файлу **/etc/shadow**, то можно считать, что пароль у него в руках. Но если следовать следующим правилам, то можно избежать взлома:

- меняйте пароли каждый месяц. Если хакер взламывает систему удаленно, то это может сделать подбор невыполнимым. Если взлом происходит локально, то пока хакер подберет пароль, он уже изменится;
- проверяйте свои пароли на стойкость от подбора по словарю. Если найдено несоответствие критерию, заставьте пользователя сменить пароль;
- устанавливайте сложные и длинные пароли, чтобы подбор по словарю стал невозможным;
- защищайте файл **/etc/shadow**. Если файл **/etc/passwd** нужен для чтения всем пользователям для нормальной работы множества утилит, то **/etc/shadow** необходимо охранять всеми возможными средствами;
- следите за журналом безопасности на предмет появления большого количества записей о неверном входе в систему.

Соблюдая эти правила, вы понизите вероятность взлома вашей системы методом перебора паролей.

В *разд. 2.6* мы говорили о необходимости создания сложных паролей и рассмотрели некоторые рекомендации по этому вопросу. А сейчас я хочу предложить вам для этого еще один интересный метод:

- создайте файл pass.txt с текстом, который нужно использовать в качестве пароля, например: `echo "password" >> pass.txt`;
- шифруем файл с помощью OpenSSL. Для этого выполняем команду: `openssl des -in pass.txt -out pass.txt.s`. Ключ, который запросит про-

грамма для шифрования, не имеет значения, можно даже использовать слово password;

- просмотрите содержимое файла pass.txt.s. В нем будет зашифрованный текст, который вы записали в файл pass.txt. Выберите читаемые символы и используйте их в качестве пароля. Такое нарочно не придумаешь, поэтому программы подбора по словарю будут бессильны, останется только полный перебор.

Отличным методом защиты от удаленного подбора может быть модульная аутентификация, которую мы рассматривали в *разд. 3.3*. Среди PAM есть очень удобный в защитных целях модуль **/lib/security/pam_tally.so**. Он позволяет блокировать доступ после определенного количества попыток входа в систему. Рассмотрим использование модуля на примере авторизации в Linux, настройки которой находятся в файле **/etc/pam.d/login**. Для ограничения попыток ввода паролей добавим в этот файл следующую строку:

```
account required /lib/security/pam_tally.so
 deny=5 no_magic_root
```

В качестве аргумента модулю передается параметр `deny`, который равен 5. Это значит, что после этого количества попыток учетная запись блокируется. Число 5 является наиболее оптимальным. Меньшие значения приведут к проблемам, когда пользователи по несколько раз ошибаются при вводе своего пароля. Ну а если пяти попыток не хватило исправиться или вспомнить пароль, то далее идет уже подбор случайным образом, что надо запретить.

14.10.4. John the Ripper

Теперь рассмотрим на практике, как происходит подбор паролей. Это необходимо, чтобы понять технологию и уметь использовать ее самостоятельно для тестирования стойкости паролей ваших пользователей.

John the Ripper — самая популярная программа для взлома паролей, которая завоевала сердца большинства хакеров и администраторов. Она поддерживает основные алгоритмы шифрования MD5, DES и Blowfish.

Чтобы подобрать пароли, для начала необходимо выполнить следующие команды:

```
unshadow /etc/passwd /etc/shadow > pass.txt
jhon -incremental pass.txt
```

С помощью первой директивы мы получаем файл pass.txt, в котором находятся соответствия имен пользователей и паролей. Этот файл можно создать и вручную, перенеся пароли из файла **/etc/shadow** в файл **/etc/passwd**, но это занятие не для слабонервных, поэтому лучше довериться программе.

Вторая команда запускает полный перебор паролей. Если у вас есть файл со словарем, то для его использования выполните следующую команду:

```
jhon -wordfile:filename pass.txt
```

В данном случае `filename` — это имя файла словаря.

В ОС Linux уже есть словарь, который находится в файле **/usr/share/dict /words**. На заре становления Интернета самый знаменитый вирус Морриса, благодаря подбору паролей по словарю, встроенному в ОС Unix (в те времена Linux еще не было), смог взломать множество систем и заразить самое большое число компьютеров для своего времени. Для того чтобы использовать встроенный словарь, выполните команду:

```
jhon -wordfile: /usr/share/dict/words pass.txt
```

На сайте **http://packetstorm.secqurity.com** есть большая коллекция словарей, которую можно использовать для тестирования своей системы на сложность паролей. Если с помощью такого лексикона вы подберете пароль для одной из своих учетных записей, то поверьте мне, то же самое сделает и хакер.

Если нажать любую клавишу, то на экране будет отображена информация о ходе подбора. Для прерывания программы необходимо нажать сочетание клавиш <Ctrl>+<C>. Для продолжения процесса подбора можно выполнить команду:

```
jhon -restore
```

Чтобы просмотреть пароли, которые подобрала программа, необходимо выполнить следующую директиву:

```
jhon -show pass.txt
```

14.11. Тюнинг ОС Linux

В течение всей книги мы говорили о безопасной и эффективной настройке ОС Linux и ее сервисов. В данном разделе подведем итог всему сказанному ранее и рассмотрим несколько новых параметров, которые могут сделать систему еще быстрее и надежнее. Эти установки уже относятся к более тонким, поэтому я оставил их напоследок.

Мы уже говорили о том, что лучшим средством повысить безопасность и производительность является загрузка только самого необходимого. От этого напрямую зависит использование памяти и нагрузка на процессор.

После того как вы определились со списком загружаемых сервисов и сократили их до минимума, необходимо позаботиться о настройке каждого из них с учетом целесообразности использования. Здесь опять вступает в дело ми-

нимизация возможностей. Например, сервис Apache загружает множество различных модулей, абсолютно ненужных для большинства сайтов.

Каждый лишний модуль — это очередной удар по производительности и безопасности. Закончив с этим, переходим к более тонким настройкам.

14.11.1. 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
```

Что представляют собой параметры, которые вы можете видеть в файле? Попробуем разобраться на примере `net.ipv4.tcp_ecn`. На самом деле это путь к файлу относительно каталога **/proc/sys**, в данном случае это файл **/proc/sys/net/ipv4/tcp_ecn**. Как видите, я просто заменил в параметре все точки на символ слэш и прибавил результат к подкаталогу **/proc/sys**. Выполните следующую команду, чтобы просмотреть содержимое файла:

```
cat /proc/sys/net/ipv4/tcp_ecn
```

В результате на экране вы должны увидеть 0 или 1. Это и есть значение параметра.

Но корректировать файл вручную нет смысла. Для изменения лучше использовать команду:

```
sysctl -w имя_параметра = значение
```

С помощью этой же команды можно просматривать значение параметров ядра:

```
sysctl имя_параметра
```

Например, следующая директива отобразит значение параметра `net.ipv4.tcp_ecn`:

```
sysctl net.ipv4.tcp_ecn
```

В результате вы увидите то же значение, что и при просмотре файла **/proc/sys/net/ipv4/tcp_ecn** напрямую. Большинство параметров имеют логический тип, т. е. могут быть равны 0 (отключено) или 1 (включено).

Рассмотрим параметры, которые нужно изменить, а если их нет в файле, то добавить:

- `net.ipv4.icmp_echo_ignore_broadcasts` — игнорировать широковещательные эхо-ICMP-пакеты (параметр включен);
- `net.ipv4.icmp_echo_ignore_all` — запретить эхо-ICMP-пакеты (значение 1). Используйте этот параметр, если не хотите связываться с сетевым экраном. Запрет эхо-пакетов уменьшит трафик, хотя и незначительно, и при этом делает неэффективными любые атаки с помощью ping;
- `net.ipv4.conf.*.accept_redirects` — разрешить принимать перенаправления маршрутизатора. В *разд. 4.5.3* мы говорили о том, что это небезопасно и может позволить хакеру обмануть маршрутизатор и прослушать трафик атакуемой машины;

Вместо символа звездочка может быть любое имя директории. Дело в том, что в каталоге net.ipv4.conf находится несколько подкаталогов — по одному для каждого сетевого интерфейса. В вашей системе должно быть как минимум 4 директории со следующим распределением содержащейся в них информации:

- all — конфигурационные файлы, которые влияют на все интерфейсы;
- default — значения по умолчанию;
- eth0 — конфигурационные файлы первой сетевой карты;
- lo — конфигурационные файлы петлевого интерфейса loopback.

Звездочка указывает на то, что параметр должен быть назначен всем интерфейсам. В большинстве случаев достаточно заменить "*" на имя директории all, но иногда приходится подставлять все существующие директории;

- `net.ipv4.conf.*.secure_redirects` — позволить принимать сообщения от шлюза по умолчанию о перенаправлении маршрутизатора. Параметр может быть включен, только если в вашей сети действительно более одного маршрутизатора, иначе лучше запретить;
- `net.ipv4.conf.*.send_redirects` — разрешить компьютеру, если он является маршрутизатором, отправлять сообщения о перенаправлении пакетов. 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. 14.12.2* мы поговорим об этом, 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. 4.13)*. 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 196608. Слишком большое значение отнимет лишнюю память и может привести к эффекту, при котором для обработки данных не хватит ресурсов сервера. Я бы оставил это значение по умолчанию;

- `net.ipv4.ipfrag_high_thresh` — определяет максимальное количество памяти (по умолчанию 262144), выделяемое для сборки фрагментированных пакетов. Если значение превышено, то ОС начинает отбрасывать пакеты. Таким образом, хакер может закидать сервер большим количеством мусорных сообщений, и тот больше не будет выполнять дефрагментацию;
- `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 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 превышается максимально допустимое количество подключений.

Параметров ядра очень много, и рассматривать все мы не будем. Советую обратиться к документации.

14.11.2. Тюнинг HDD

Долгое время в ОС Linux для доступа к жесткому диску была отключена даже поддержка DMA (Direct Memory Access, прямой доступ к памяти), хотя эта возможность существует почти во всех материнских платах еще со времен первых компьютеров с процессором Pentium. ОС не использовала DMA в целях совместимости с более старыми компьютерами, поэтому функцию приходилось включать самостоятельно.

В современных дистрибутивах поддержка DMA уже включена, но работу винчестера можно еще оптимизировать. Для тестирования и настройки жесткого диска используется утилита hdparm. Для определения скорости работы диска выполните команду с ключом `-t`:

```
hdparm -t /dev/hda
```

В ответ вы получите сообщение типа:

```
Timing buffered disk reads: 64 MB in 3.02 seconds = 21.19MB/sec
```

Попробуйте в качестве параметра указать раздел:

```
hdparm /dev/hda2
```

В результате будут выведены параметры жесткого диска:

```
/dev/hda2:
multcount  = 128 (on)
```

```
IO_support  = 0 (default 16-bit)
unmaskirq   = 0 (off)
using_dma   = 1 (on)
keepsettings = 0 (off)
readonly    = 0 (off)
readahead   = 8 (on)
geometry    = 2088/255/63, sectors = 32515560, start = 1028160
```

В этой информации есть много интересного:

- `multcount` — количество слов, читаемых за один такт. Эта опция должна быть включена, и желательно установить значение 128. Это может повысить производительность на 30—50 %. Для изменения значения используется ключ `mX`, где `X` — это устанавливаемое значение;
- `using_dma` — режим DMA. Для включения используется ключ `d1`;
- `IO_support` — режим доступа к диску. По умолчанию стоит 16-битный, но сейчас уже можно использовать 32-битный режим. Для включения используется ключ `c3`.

Это основные три параметра, которые могут реально повысить производительность. Итак, давайте установим значения в соответствии с указанными выше рекомендациями. Для этого выполните команду:

```
hdparm -m128d1c3/dev/hda
```

Как видите, мы просто перечислили 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`.

Помимо того, что вы увидели с помощью команды `hdparm /dev/hda2`, появился еще один параметр — режим доступа. В настоящее время поддерживается три режима ATA 33/66/100. Сверьтесь с документацией на жесткий диск, чтобы узнать, что он поддерживает.

Для смены режима используется ключ `X`:

- X34 — ATA33;
- X68 — ATA66;
- X69 — ATA100.

Для установки ATA66 выполните команду:

```
hdparm -X68/dev/hda
```

Самое странное, что 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
```

14.11.3. Автомонтирование

Если вы начали знакомство с компьютером под ОС Windows, то для вас может оказаться дикостью процесс ручного монтирования файловых систем и особенно CD-ROM-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/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 **/etc/auto.master**. Просмотрите его содержимое в листинге 14.2.

Листинг 14.2. Конфигурационный файл /etc/auto.master

```
# $Id: auto.master,v 1.2 1997/10/06 21:52:03 hpa Exp $
# Sample auto.master file
# пример файла auto.master
# Format of this file:
# формат этого файла:
# mountpoint map options
# точка_монтирования карта настройки
# For details of the format look at autofs(8).
# для дополнительной информации выполните
# команду man autofs
/misc    /etc/auto.misc    --timeout=60
```

Если не считать комментариев, в этом файле только одна строка, которая может быть полезной — последняя. В вашей системе она может быть закомментирована, но для использования автоматического монтирования необходимо убрать знак "#".

У конфигурационной строки следующий формат:

```
точка_монтирования карта настройки
```

В данном случае точкой монтирования выступает директория **/misc**. Это немного затрудняет работу, потому что при ручном подключении используется директория **/mnt**. Второй параметр определяет карту монтирования. В данном случае это файл **/etc/auto.misc**. Формат и назначение файла чем-то похожи на файл **/etc/fstab**, который используется для команды `mount`. Содержимое файла **/etc/auto.misc** можно увидеть в листинге 14.3.

Последний параметр `--timeout=60`, это время простоя. Если в течение этого периода в директории, использованной под подключение, не будет активности, то устройство будет размонтировано. По умолчанию установлено 60 секунд. В большинстве случаев это вполне приемлемое значение.

Листинг 14.3. Содержимое файла /etc/auto.misc

```
# $Id: auto.misc,v 1.2 1997/10/06 21:52:04 hpa Exp $
# This is an automounter map and it has the following format
# Это карта автомонтирования, которая имеет следующий формат:
# key [ -mount-options-separated-by-comma ] location
# Details may be found in the autofs(5) manpage
# Дополнительную информацию можно получить в man autofs

cd          -fstype=iso9660,ro,nosuid,nodev       :/dev/cdrom

# the following entries are samples to pique your imagination
# следующие записи являются примерами для
# пробуждения воображения
#linux              -ro,soft,intr     ftp.example.org:/pub/linux
#boot               -fstype=ext2              :/dev/hda1
#floppy             -fstype=auto              :/dev/fd0
#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 CD-ROM диска:

```
cd    -fstype=iso9660,ro,nosuid,nodev    :/dev/cdrom
```

Первый параметр определяет директорию внутри **/misc**, куда будет монтировано устройство. Второй атрибут — параметры файловой системы и ключи, которые будут использоваться для подключения. В случае с CD-ROM указана файловая система iso9660, разрешено только чтение и запрещены SUID- и DEV-программы. Последний параметр определяет устройство, которое должно монтироваться.

Как видите, все очень даже просто. Если попытаться обратиться к директории **/misc/cd**, и в приводе CD-ROM в этот момент будет находиться диск, то он будет автоматически смонтирован. Только есть одно замечание, адресоваться лучше командами Linux (например, выполнить команду `ls /misk/cd`), а не с помощью программ. Если просмотреть директорию **/misc/cd** с помощью Midnight Commander, то диск не будет смонтирован.

14.12. Короткие советы

Мы проанализировали достаточно много аспектов создания безопасной системы, но есть некоторые общие рекомендации, которые подытоживают рассмотренный в этой книге материал. Поэтому напоследок я собрал короткие советы, которые пригодятся вам в построении безопасного сервера или сети.

14.12.1. 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
```

14.12.2. Маршрутизация от источника

Мы уже не раз говорили о том, как пакеты проходят по сети, и достаточно подробно затронули эту тему в *разд. 4.5.1*. Напоминаю, что внутри сети пакеты передаются по MAC-адресу, а для обмена между сетями необходимо маршрутизирующее устройство, которое умеет работать с IP-адресами и направлять пакеты по нужному пути, который они сами определяют. Но эти устройства управляемы, и существует несколько методов заставить их устре-

мить пакеты в нужное русло. Один из этих методов — маршрутизация от источника (source routing).

С помощью маршрутизации от источника можно установить путь прохождения пакета по сети. Иногда это действительно удобно, но мы же знаем, что это "палка о двух концах", и вполне логичным было бы source routing запретить, а лучше вообще никогда не придумывать.

Как маршрутизация от источника влияет на безопасность? Допустим, что ваш сетевой экран запрещает подключения с адреса 192.168.1.1, потому что через этот IP к вам пытался проникнуть хакер. Так как все пакеты злоумышленника маршрутизаторы направляют именно через этот адрес, то подключение становится невозможным. Но благодаря source routing злоумышленник может сам указать путь, по которому должен следовать пакет, и провести его в обход маршрутизатора или сервера с запрещенным адресом.

Жаль, что мы не можем запретить маршрутизацию от источника на компьютере хакера, но мы не должны принимать такие пакеты на своей стороне и тем более на компьютере, который выполняет роль шлюза в Интернет (прокси-сервер или сетевой экран). Для этого необходимо установить 1 в файле **/proc/sys/net/ipv4/conf/all/accept_source_route** или выполнить команду:

```
echo 0 > /proc/sys/net/ipv4/conf/all/accept_source_route
```

14.12.3. SNMP

Протокол SNMP (Simple Network Management Protocol, простой протокол сетевого управления) применяется для управления сетевыми устройствами, такими как маршрутизаторы, управляемые коммутаторы и даже бытовыми устройствами, подключенными к сети.

Существует три версии этого протокола. Первая версия была разработана очень давно и, конечно же, осуществляла открытый обмен паролями и данными. Шифрование было добавлено в SNMP, начиная со второй версии. Именно поэтому первую версию не рекомендуется использовать, а лучше даже запретить.

У SNMP есть еще один недостаток — протокол использует в качестве транспорта UDP-протокол, который не поддерживает виртуальное соединение, и передача осуществляется простой отправкой пакетов в сеть без подтверждения доставки и без каких-либо авторизаций, т. е. злоумышленник легко может подделать любые поля пакета.

Я не рекомендую использовать SNMP вообще, потому что для большинства задач можно обойтись без него. Конечно же, шифрование, добавленное во второй версии, значительно повысило безопасность, и его можно стало применять даже в особо важных случаях. Но необходимо сначала убедиться, что

вы работаете именно со второй или более старшей версией и шифрование используется.

14.12.4. 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`:

- хакер создает в каком-либо каталоге (например, в общедоступном **/tmp**) файл с таким же именем, как и у атакуемой программы;
- в этот файл может быть записан сценарий, который выполняет необходимые хакеру действия.

Например, следующий код может быть записан в такой файл:

```
#!/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 данном примере выполняется 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 ls права, которые позволят выполнять ее любому пользователю:

```
chmod 777 /tmp/ls
```

Ложный файл готов. Теперь необходимо сделать так, чтобы он выполнялся вместо системной команды `ls`. Для этого достаточно добавить в системную переменную окружения `PATH` в самое начало каталог **/tmp**. Если теперь кто-либо запустит команду `ls` без указания полного пути, то выполнится наш сценарий, который попытается изменить права доступа на файлы паролей. Если у пользователя, запустившего команду, хватит полномочий для изменения прав, то можно считать, что система взломана.

Следите за содержимым системной переменной окружения PATH, чтобы ее никто не изменил.

14.12.5. Доверительные хосты

В файле **.rhosts** можно прописать адреса хостов, которым вы доверяете. Пользователи этих компьютеров смогут подключаться к серверу без аутентификации с помощью таких программ, как Telnet или FTP.

Мы уже столько раз говорили о безопасности, что нетрудно догадаться, что хакер может подделать адрес отправителя. Если ему удастся это сделать, то ваш сервер превращается в проходной двор.

14.12.6. Защита паролей

Для защиты паролей Linux достаточно только охранять файл **/etc/shadow**. Помимо этого вы должны контролировать сложность паролей, регулярно пытаясь подобрать пароль по популярным словарям, которые легко найти в Интернете (*именно их используют хакеры*). *Если пароли сложные*, то даже при получении файла взломщику понадобится слишком много времени, и скорей всего ничего не удастся.

Но не все так просто. Если пароли для доступа к системе защищаются самой ОС и обязательно шифруются, то остальные программы могут не иметь такой возможности. Например, в пользовательских программах для доступа к определенным сервисам, в частности FTP или POP3, может не использоваться шифрование, и в этом случае пароли могут находиться в конфигурационном файле в открытом виде.

Прежде чем устанавливать какую-либо программу, узнайте, где она хранит пароли и как они защищены (используется шифрование или нет). Устанавливайте такие права на файлы, чтобы доступ к ним мог получить только конкретный пользователь и администратор. Желательно, чтобы для группы стояли нулевые права, особенно если в ней может быть несколько пользователей.

Если отдельная группа создается для каждого пользователя, то группе можно выделить некоторые права. И все же, я бы не советовал этого делать, потому что неизвестно, что будет в будущем. Хакер может добавить себя в определенную группу, или вы сами можете случайно или намеренно объединить пользователей.

Всем своим пользователям я рекомендую не сохранять в программах паролей. Это значит, что при проверке почты надо каждый раз указывать свой пароль. Но это очень неудобно, особенно когда у пользователя более трех почтовых ящиков, а это в наше время является нормой. Да и при наличии

одного очень трудно заставить пользователя помнить все пароли и не сохранять их в системе.

Но вводить пароль надо непосредственно в программе, и лучше всего, если он не будет отображаться на экране. Это значит, что надо стараться не указывать пароли в командной строке, которая не может использовать невидимые символы.

Существует множество методов, с помощью которых можно увидеть набираемый пароль, например, команда ps. Пример правильного получения пароля — утилита login, которая не отображает на экране вводимые данные.

В открытом виде пароли могут храниться и в базах данных, именно там содержится наиболее важная информация для любой организации. Базы данных требуют отдельного разговора, и в данной книге мы эту тему не затрагиваем, но забывать о ней нельзя.

14.12.7. Перенаправление сервисов

Если существуют какие-либо ресурсы, к которым обращается ограниченный круг людей, то необходимо заставлять сервисы работать на нестандартных портах. Это позволит защитить систему от излишних посягательств.

Одной из распространенных проблем использования стандартных портов является возможность их сканирования. Например, хакер узнает об уязвимости БД определенного разработчика. Допустим, что эта база использует порт 1457. Злоумышленнику достаточно только запустить сканирование сети в поисках компьютера с открытым портом 1457. Как только машина найдена, она попадает в "черный" список, и хакер может запустить программу, которая автоматически захватит все компьютеры из списка.

Проблема легко решается переконфигурированием сервиса для работы на другом порту, и при этом необходимо убрать любые баннеры, которые появляются при подключении на этот канал. В этом случае хакер не сможет узнать, что работает на порту и как взаимодействовать с удаленной программой.

Если с сервисами сервера работает небольшое количество людей, то можно поменять местами особо уязвимые (те программы, которые предоставляют пользователям возможность записи или выполнения команд в системе), например FTP (заставить работать на 80 порту) и HTTP (настроить на 21 порт). Жаль, что это нельзя сделать с общедоступными сервисами, а если и можно, то бесполезно. Например, если заставить работать Web-сервер на 81 порту вместо 80, то любой пользователь Интернета должен иметь возможность узнать об этом. А тогда и хакер будет в курсе.

14.13. Обнаружен взлом

Если вы обнаружили, что в системе есть посторонний, а сервер содержит секретную информацию, потеря которой может оказаться фатальной для вас, я рекомендую остановить сетевой интерфейс, чтобы компьютер отключился от сети, и начать анализ системных журналов. Лучше пусть сервер будет полчаса недоступным, чем вы полностью потеряете над ним контроль.

Для анализа первым делом необходимо запустить проверку конфигурации системы (об этом мы говорили в *разд. 12.3*). Необходимо сравнить отчеты программ проверки до и после взлома. Это поможет вам понять, что успел сделать хакер. Если в вашей системе оказался rootkit, то его нужно удалить.

Следующим этапом необходимо проверить контрольные суммы всех основных файлов, особенно конфигурационных из каталога **/etc** и исполняемых из каталога **/bin**. Злоумышленник может изменить эти файлы, чтобы получить потайной вход и оставаться в системе незамеченным. Определив отклонения, постарайтесь вернуть все в исходное состояние.

Далее, анализируем целостность пакетов. Сначала выполняем команду:

```
rpm -qa | grep kernel
```

Таким образом можно проверить установленные пакеты ядра, а потом обследовать их. Если вы увидели изменения, то добейтесь возврата в исходное состояние.

После этого необходимо проверить обновления для ОС Linux и используемых вами служб. В большинстве случаев взлом происходит именно из-за устаревшего программного обеспечения. Обновите все программы. Не забудьте и Web-сценарии, которые используются на Web-сервере, потому что они также нередко становятся причиной взлома.

Если у вас Web-сервер, то я не спешил бы возобновлять его работу, потому что это может быть опасно. Возможно, что ошибка еще не исправлена, если у вас много сценариев, написанных самостоятельно. Если ваш компьютер выполняет функции почтового сервера или решает любые другие задачи, то можно возобновить работу, но продолжать тщательное наблюдение за системой.

И только на последнем этапе я рекомендую начать анализ журналов и выявление действий хакера, которые привели к взлому, параллельно наблюдая за работающей системой. Если злоумышленник снова попытается проникнуть, то вы должны увидеть это и предотвратить вторжение на раннем этапе, чтобы не пришлось повторять процесс чистки системы.

Пока вы анализируете журналы, все пользователи должны сменить свои пароли доступа к серверу и ко всем его службам.

Разбирая журналы, вы должны определить:

- какие службы использовал хакер, и в каких журналах есть записи о его активности на вашем сервере;
- какими учетными записями смог воспользоваться взломщик;
- какие команды он выполнял.

Вы должны узнать как можно больше, чтобы определить, было ли выполнено все необходимое для предотвращения повторной попытки взлома. Некоторые администраторы просто возобновляют работу сервера и через какое-то время расплачиваются за это.

Желательно получить максимальное количество информации о хакере, чтобы эти сведения можно было предоставить в правоохранительные органы. Не пытайтесь бороться со взломом самостоятельно, потому что у вас может не хватить сил. Обратитесь за помощью в компетентные организации, которые обладают достаточными ресурсами для выявления и пресечения атак. Но, чувствуя безнаказанность, хакер будет продолжать вторжения, и может возникнуть ситуация, когда взломщик успеет получить свое.

ПРИЛОЖЕНИЕ 1

Команды FTP-протокола

Когда вы подключаетесь к FTP-серверу с помощью клиента, работающего из командной строки (например, sftp, рассмотренный в *разд. 5.3*), то для работы с сервером вам понадобятся следующие команды:

- `cd путь` — изменить текущую директорию на указанную. Чтобы подняться на один уровень выше, можно выполнить команду `cd ..`, а чтобы перейти в папку ниже текущего уровня, то можно выполнить команду `cd директория`;
- `bye` — разорвать соединение;
- `exit` — выйти из системы;
- `chmod права имя файла` — изменить права доступа к файлу. Например, если вы хотите установить права доступа 770 на файл passwd в текущей директории, то нужно выполнить команду `chmod 770 passwd`;
- `get -P удаленный_файл локальный_файл` — скачать файл. Ключ `-P` является необязательным и 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 скаченный с сервера файл;
- `put -P локальный_файл удаленный_файл` — закачать локальный файл на сервер;
- `help` — отобразить список команд, которые разрешено выполнять;
- `pwd` — показать текущую директорию;
- `rm файл` — удалить файл;
- `rmdir директория` — удалить директорию;
- `mkdir имя` — создать директорию с указанным именем.

Вы должны учитывать, что разные FTP-серверы и FTP-клиенты могут обрабатывать команды по-разному.

ПРИЛОЖЕНИЕ 2

Полезные программы

- hunt (**http://lin.fsid.cvut.cz/~kra/index.html**) — одна из популярных программ прослушивания трафика. В качестве дополнительных возможностей в нее встроены: подделка MAC-адресов через рассылку фальшивых ARP-пакетов и перехват соединения;
- dsniff (**http://monkey.org/~dugsong/dsniff/**) — пакет программ для прослушивания трафика, который состоит из следующих утилит:
  - dsniff — перехват паролей (основная задача). Она прослушивает трафик в ожидании пакетов авторизации, и если такой пакет найден, то программа выводит на экран заветный пароль. Поддерживается поиск пакетов авторизации всех основных протоколов, таких как Telnet, FTP, POP и т. д.;
  - arpspoof — позволяет отправлять ARP-ответы, с помощью которых можно обмануть компьютеры жертвы, сказав, что определенный IP-адрес принадлежит вам;
  - dnsspoof — позволяет отправлять поддельные DNS-ответы. Если жертва запрашивает IP-адрес сервера, вы можете подменить ответ DNS-сервера, чтобы вместо него клиент подключился к вам, и вы смогли перехватить на себя трафик;
  - filesnaf — прослушивает трафик в ожидании передачи файлов по протоколу 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;
  - macof — позволяет наводнить сеть пакетами, включающими сгенерированные MAC-адреса. Если коммутатор не справляется 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 с помощью отправки пакета с установленным флагом RST;
- webspy — позволяет прослушивать соединения с Web-серверами и сохраняет список сайтов, которые посещал определенный пользователь;
- webmint — эмулирует Web-сервер и становится посредником. Технология атаки этим методом рассмотрена в *разд. 7.9*;

- ettercap (**http://ettercap.sourceforge.net**) — на мой взгляд, это самая удобная программа для прослушивания трафика. Основное ее назначение — поиск паролей в пакетах всех популярных протоколов. Администраторы оценят возможность определения наличия в сети других прослушивающих программ;
- lsat (**http://usat.sourceforge.net/**) — программа проверки конфигурации системы (рассматривается в *разд. 12.3*). Анализирует настройки сервера, отображая потенциальные ошибки, и в некоторых случаях может дать рекомендации по устранению недочетов;
- bastlle (**http://bastille-linux.sourceforge.net/**) — выявление потенциальных ошибок в конфигурации сервера. Программа может автоматически исправлять ошибки и недочеты в конфигурации;
- Klaxon (**http://www.eng.auburn.edu/users/doug/second.html**) — программа позволяет определить атаки на вашу систему (рассматривается в *разд. 12.4*);
- PortSentry (**http://sourceforge.net/projects/sentrytools**) — следит за портами и отслеживает попытки их сканирования (рассматривается в *разд. 12.4*). Позволяет автоматически сконфигурировать сетевой экран для запрета соединения с компьютером, с которого происходило сканирование;
- Swatch (**http://sourceforge.net/project/swatch**) — удобная программа анализа по расписанию системных журналов (рассматривается в *разд. 12.6*);
- Logsurfer (**sourceforge.net/projects/logsurfer**) — одна из немногих программ, позволяющих анализировать журнал безопасности в динамике (рассматривается в *разд. 12.6*);
- John the Ripper (**http://www.openwall.com/john/**) — самая знаменитая программа подбора паролей;
- pop-before-smtp (**http://popsmtp.sourceforge.net**) — сервис, который разрешает отсылку писем только в том случае, если пользователь сначала проверил почту по POP3-протоколу;
- Nmap (**www.insecure.org/nmap/**) — сканер портов, который обладает большим количеством возможностей.

ПРИЛОЖЕНИЕ 3

Интернет-ресурсы

- **http://www.redhat.com** — сайт компании Red Hat, где можно скачать последние версии программ, ядра, патчи и прочитать о перспективах развития ОС;
- **http://www.kernel.org** — сайт, посвященный ядрам ОС Linux. Здесь же можно скачать последнюю версию ядра;
- **http://www.securityfocus.com** — сайт, посвященный безопасности, содержит множество описаний различных уязвимостей и методов исправления ошибок;
- **http://www.cert.org** — еще один сайт, посвященный информационной безопасности;
- **http://www.securitylab.ru** — русскоязычный сайт по информационной безопасности;
- **http://www.xakep.ru** — сайт Российского журнала "Хакер";
- **http://www.insecure.org** — множество полезной информации по безопасности, статьи и программы;
- **http://www.novell.com/de-de/linux/suse/** — сайт SUSE, один из самых простых и удобных дистрибутивов Linux;
- **http://www.linspire.com/** — разработчики Linspire пытаются создать ОС, в которой могли бы выполняться программы Windows на ядре Linux;
- **http://www.debian.org** — официальный дистрибутив Debian;
- **http://www.slackware.com** — дистрибутив SlackWare.

Источники информации

1. Фленов М. Е. Программирование на C++ глазами хакера. — СПб.: БХВ-Петербург, 2004.
2. Фленов М. Е. Программирование в Delphi глазами хакера. — СПб.: БХВ-Петербург, 2003.
3. Фленов М. Е. Компьютер глазами хакера. — СПб.: БХВ-Петербург, 2005.
4. **http://www.vr_online.ru** — сайт для программистов и администраторов.
5. **www.xakep.ru** — сайт журнала хакер.

Предметный указатель

H

I

K

L

M

N

O

P

R

S

T

U

W

X

A

Б

B

Г

Д

Ж

З

И

К

Л

М

Н

О

П

P

С

T

У

Ф

Ч

Ш

Я