Михаил Фленов

Linux
ГЛАЗАМИ
ХАКЕРА
3-е издание

Санкт-Петербург

«БХВ-Петербург»

2010

УДК 681.3.06
ББК 32.973.26-018.2
Ф69

Фленов М. Е.

Ф69 Linux глазами хакера: 3-е изд., перераб. и доп. — СПб.: БХВ-Петербург, 2010. — 480 с.: ил. + CD-ROM

ISBN 978-5-9775-0547-5

Рассмотрены вопросы настройки ОС Linux на максимальную производительность и безопасность. Описаны базовое администрирование и управление доступом, настройка Firewall, файлообменный сервер, WEB-, FTP- и Proxy-серверы, программы для доставки электронной почты, службы DNS, а также политика мониторинга системы и архивирование данных. Приведены потенциальные уязвимости, даны рекомендации по предотвращению возможных атак и показано как действовать при атаке или взломе системы, чтобы максимально быстро восстановить ее работоспособность и предотвратить потерю данных. В третьем издании материал переработан и дополнен новой информацией в соответсвии с современными реалиями. На компакт-диске находятся дополнительная документация и программы в исходных кодах.

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	*Юрий Якубович*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Наталия Першакова*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 30.04.10.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 38,7.
Тираж 2500 экз. Заказ №
"БХВ-Петербург", 190005, Санкт-Петербург, Измайловский пр., 29.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.60.953.Д.005770.05.09 от 26.05.2009 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0547-5

Оглавление

Предисловие

Эта книга посвящена рассмотрению одной из самых популярных операционных систем (ОС), устанавливаемых на серверы, — ОС Linux. Для домашнего использования эта система пока еще не получила такой же популярности, как среди профессиональных администраторов, но в последнее время наметились предпосылки для захвата и этого рынка. ОС Linux становится все проще, удобнее и красивее. Единственное, чего не хватает этой системе (на мой взгляд) — такого же количества игр, как для платформы Windows.

Установка ОС Linux становится проще, а графический интерфейс и удобство работы в некоторых случаях не уступает самой распространенной в среде малого бизнеса ОС Windows. Есть еще некоторые шероховатости, которые не позволяют ОС Linux получить широкое распространение среди домашних пользователей, но мы только рады этому. Конкуренция заставляет компании развиваться и улучшать свои продукты, и это развитие заметно даже без бинокля.

Эта книга будет полезна администраторам Linux и тем пользователям, которые хотят познакомиться с этой системой поближе. Рассматриваемые вопросы настройки и безопасности пригодятся специалистам по безопасности сетей, использующих различные ОС, потому что значительная часть приводимой информации не привязана к определенной системе, а теория защиты различных систем похожа.

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 можно проникнуть 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. Но, несмотря на то, что книга может послужить отправной точкой для хакера, я надеюсь, что этого не произойдет. Помните о законности ваших действий.

Я интересуюсь взломом и постоянно изучаю новые методы, но только потому, что я хочу строить безопасные системы, и безопасность меня интересует намного больше. Любой объект может быть рассмотрен с разных точек зрения. Простой пример из жизни: нож, являясь столовым прибором, при определенных обстоятельствах становится орудием убийства или средством самообороны. Точно так же и методы хакеров, которые будут рассматриваться в этой книге, могут быть восприняты как советы для повседневного ухода за ОС и способы защиты от проникновения или же как средства взлома системы. Я надеюсь, что вы не будете использовать полученные знания в разрушительных целях. Это вас не украсит и не добавит ума. Зачем вам нужна "черная" популярность взломщика? Не лучше ли посвятить себя более полезным и добрым вещам?

Несмотря на явное стремление Linux поселиться в домашних компьютерах, настройка этой ОС пока еще слишком сложна. Для того чтобы правильно это сделать, нужно знать множество параметров, которые большинству пользователей не нужны. Если же просто закрыть глаза и оставить все значения по умолчанию, то об истинной безопасности Linux не может быть и речи. Ни одна ОС не может работать надежно и с максимальной защитой при настройках по умолчанию. Дело в том, что производитель не может заранее знать, что именно нам понадобится, и делает все возможное, чтобы программа работала на любой системе, а для этого приходится включать много дополнительных возможностей, что делает систему избыточной. В последнее время разработчики дистрибутивов и других серверных программ стали максимально урезать установки по умолчанию, то есть разрешать только базовые возможности, а все сетевые сервисы, которые могут позволить хакеру проникнуть на компьютер, отключать. При этом чаще всего производитель предоставляет нам простое и удобное средство для быстрого включения и конфигурирования нужного сервиса.

Так уж повелось, что администраторы Linux должны иметь больше опыта и знаний, чем специалисты Windows, и это связано как раз со сложностями настройки. В этой книге я постарался максимально доступно рассказать вам про ОС Linux.

Почему книга называется "Linux глазами хакера", и что это за глаза? Этот вопрос интересует многих моих читателей. Когда мы берем в руки какую-

нибудь вещь, то надеемся, что ее внешний вид соответствует внутреннему содержанию. В данном случае вопрос в том, какая информация будет отвечать этому названию? Для ответа на этот вопрос необходимо четко понимать, кто такой хакер, и что он видит в операционной системе.

Когда меня спрашивают, что я подразумеваю под словом "хакер", я привожу простейший пример: если как администратор вы установили и заставили работать ОС, и вам удалось настроить ее на максимальную производительность и безопасность, то вы — хакер. Умения хакера позволяют создавать что-либо, превосходящее свои аналоги (то есть более быстрое, удобное и безопасное). Именно такой является сама ОС Linux, созданная хакерами со всего мира.

Данная книга рассматривает ОС, начиная с самых основ и заканчивая сложными манипуляциями с 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 может потребоваться только хорошее знание английского языка, которое позволяет читать документацию или файлы HOWTO, поставляющиеся с системой Linux.

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

Описание утилит для администрирования Linux всегда можно найти в Интернете или в документации на ОС, а вот информацию по эффективному их 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. Здесь же мы больший упор делаем на определенную ОС — Linux. Несмотря на то, что книга [3] направлена в большей степени на поддержание безопасности ОС Windows, многие рассматриваемые в ней проблемы могут вам пригодиться и при построении безопасного 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. О вирусных атаках и возможностях их отражения можно также прочитать в книге "Компьютер глазами хакера" [3].

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

QualitySource

Мой взгляд на Linux может вам понравиться, а может и шокировать. Дело в том, что я не принадлежу к сторонникам или поклонникам Open Source, к которому относится Linux. Я являюсь сторонником движения QualitySource, то есть качественного кода. Мне все равно, какой это код — открытый или закрытый, главное, чтобы он был качественный. Если бы код ОС Windows был открытым, вы бы полезли его смотреть или изменять? Я бы нет, и большинство тоже.

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

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

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

Программа не может быть лучше, надежнее или безопаснее других только потому, что у нее открыт код, это бред полнейший. Яркий пример — sendmail.

Бездарных программ с открытым кодом очень много, как и коммерческих, поэтому выбирайте за качество, а не за наличие или отсутствие исходных кодов, которые большинству пользователей просто не нужны.

Второе издание

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

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

Третье издание

В третьем издании я уже в основном обновлял информацию в соответствии с современными реалиями. Компьютерный мир изменяется очень быстро, за что я его и люблю, потому что приходится постоянно изучать что-то новое. Очень много новой информации попало на компакт-диск к этой книге в виде текстовых файлов.

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

Благодарности

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

Первым делом я хотел бы поблагодарить издательство "БХВ-Петербург", с которым работаю уже несколько лет. Спасибо руководству издательства, редакторам и корректорам, которые работают со мной и помогают сделать книгу такой, какой я ее задумывал. Ведь писать приходится в тяжелых по срокам условиях, но иначе нельзя, так как 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 благодарность моим друзьям и знакомым, которые что-то подсказывали, помогали идеями и программами.

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

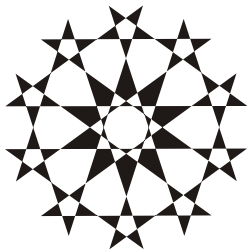
Хочется поблагодарить еще одного кота, который является ассистентом в пакете программ MS Office. Книгу я писал в MS Word, а ОС Linux работала в виртуальной машине, чтобы можно было делать снимки экрана. Во время написания первого издания, если на меня бросали ребенка, то кот-ассистент помогал занять 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 моем блоге **www.flenov.info**. Последние работы основываются на их вопросах и предложениях. Если у вас появятся какие-то проблемы, то милости прошу на сайт. Я постараюсь помочь по мере возможности, и жду любых комментариев по поводу этой книги. Ваши замечания помогут мне сделать ее лучше.

В последнее время в России перестают читать книги, качая из Интернета пиратские копии, нанося ущерб авторам и издательствам. Доход от книг падает, и многие хорошие авторы перестают писать, ибо хорошему специалисту просто найти хороший источник дохода без лишних мучений. От этого количество хороших книг уменьшается. Боюсь, эта тенденция сохранится. Если вам нравится книга, купите ее бумажный вариант, не портите глаза, читая с монитора. Здоровье важнее. Ну а я со своей стороны постарался наполнить книгу и прилагаемый компакт-диск максимальным количеством полезной информации. На компакте вы найдете множество дополнительных документов.

Если вы нашли какую-то ошибку в книге, просьба сообщить мне об этом через обратную связь на моем сайте **www.flenov.info**. Я также жду ваших мнений о книге и пожеланий о том, что вы хотите увидеть в будущем, если появится новое издание.

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



Глава 1

Прежде чем начать...

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 сервер Samba, 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, и лишь когда их систему взламывают, начинают задавать вопросы: "А почему Microsoft не дала нам нужных инструментов, чтобы запретить определенные действия?"

Простота настройки сетевого окружения в Windows 9*x* и сложность настройки в Linux связана с безопасностью. В Windows версий 9*x* не было вообще никакой защиты и разграничений доступа, только банальный пароль. Начиная с Windows 2000 настройка усложнилась, потому что теперь автоматически устанавливаемого всеобщего доступа к компьютеру нет, и теперь обе системы находятся в одинаковых условиях. Теперь можно и нужно давать права доступа только к тем ресурсам, которые реально нужны пользователю.

Если смотреть на ОС Linux с точки зрения пользователя, то после установки системы 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 должно. По умолчанию в системе должны быть запрещены практически все действия, которые могут привести к нежелательному результату или вторжению по сети. Для изменения ограничений нужно настраивать конфигурационные файлы, редактировать которые крайне неудобно, 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. И ему пришлось делать схожие вещи, потому что эта служба по умолчанию не устанавливается с ОС, и прежде чем она начнет работать, ее нужно 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 и безопаснее, а большинство удобных функций, которые могут нарушить защиту, просто отключаются. При необходимости их приходится включать. Начиная с 2008 года эта тенденция приняла совершенно новый и интересный оборот — в Windows Server появилась версия без графического режима. Да, Windows запускается в текстовом режиме, в котором можно полноценно управлять сервером! В Linux все было наоборот, эту ОС создавали с точки зрения "лишь бы было безопасно", а теперь двигаются в сторону наращивания и упрощения сервисов.

Удобство и безопасность во многом противоречат, поэтому производителю приходится чем-то жертвовать. Что мне не нравится в некоторых дистрибутивах Linux — если установить какой-либо сервис, то инсталлятор мало того, что устанавливает сервис в максимальной конфигурации, он еще и запускает его автоматически при старте системы. Это очень плохо, и такие вещи нужно пресекать.

1.1. Что такое Linux?

ОС Linux — это свободная операционная система, исходные коды которой открыты для всеобщего просмотра и даже для внесения изменений. Большинство не смотрит на исходные коды, но воспринимает их наличие, как дополнительную приятную халяву. Но даже халява должна быть полезной, а если она бесполезна, то только мусорит.

Нет, наличие исходных кодов — это хорошо, и я тут не спорю. Это преимущество для такого продукта, как ОС, потому что можно перекомпилировать ядро и максимально оптимизировать его работу именно под ваше железо. Если для пользовательских утилит перекомпиляция не нужна и, скажем, текстовый редактор можно оптимизировать лишь незначительно, то для ОС возможность компиляции исходных кодов — большой и жирный плюс, который

позволяет выжать из компьютера максимум. Но в ОС Linux есть много других преимуществ, куда более важных.

К чему я это говорю? К тому, что я люблю ОС Linux не за халяву и не за наличие исходных кодов, а за качество. Я буду любить, даже если исходные коды закроют или будут взимать плату.

Основа ядра ОС была создана в 1991 году студентом хельсинкского университета (University of Helsinki) по имени Линус Торвальдс (Linus Torvalds). Все начиналось с того, что Линус начал писать программу терминала, функции которого постепенно стали выходить за пределы простой коммуникации. Он написал костяк, функционально схожий с UNIX-системами, сделал его доступным для всеобщего просмотра и доработки и обратился с просьбой помогать ему в улучшении и наращивании возможностей новой системы. 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 рассматривают как врага. В любом случае равнодушных очень мало.

Официальная версия ядра ОС под номером 1.0 была выпущена в 1994 году, то есть 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 такую популярность, ведь были и есть другие открытые проекты, некоторые из которых 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.

Linux — не единственная свободная система и не единственная ОС, исходные коды которой открыты. Но, несмотря на это, именно она стала популярной. ОС Linux не идеальна, и я не буду говорить, что она лучшая, но, все же, она покорила сердца миллионов, а может даже миллиардов, людей. Если учесть, что Linux является чуть ли не официальной ОС китайцев, то если миллиарда пользователей еще нет, то он точно появится.

Почему мы влюбляемся во что-то или в кого-то? Вот сижу я сейчас, смотрю на свою жену, и не могу ответить на этот вопрос однозначно. Наверное, про-

сто любим! Так же я не могу объяснить, почему люблю Windows или Linux. Я не могу понять, почему я сегодня предпочитаю KDE, а завтра загружаю GNOME.

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. Используя Linux, многие фирмы смогли оптимизировать свои расходы на инфраструктуру информационных технологий (ИТ) за счет перевода некоторых серверных задач на бесплатный продукт — Linux. А о домашних пользователях я вообще молчу, ведь для них полноценный UNIX был слишком дорог.

Среди первых задач, которые стали доверять Linux, — организация WEB-сервера, и с ней эта ОС справляется великолепно. Трудно точно оценить, какой процент хостинговых компаний сейчас использует Linux, но большинство статистических анализов показывает, что на Linux совместно с сервером Apache приходится большая часть, по некоторым данным, более половины. Но тут заслуга не только Linux, но и WEB-сервера Apache, который также бесплатен, надежен и просто великолепен.

На данный момент в Linux можно сделать практически все. Для этой ОС уже написано 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 несколько 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, хотя европейский суд и пытается добиться этого. В Linux конкуренция действительно свободная, никто не запрещает использовать сторонние разработки и не борется с этим. Напротив, пользователю предоставляется реальный выбор.

1.2. Открытый исходный код — безопасно?

Бытует мнение, что программы с открытым исходным кодом надежнее и безопаснее, чем фирменные. Сторонники этого утверждения считают, что такую систему исследуют множество людей разными способами и тем самым

выявляют все возможные погрешности. 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" зачастую демонстрирует совершенно непредсказуемые результаты.

К тому же, сколько бы человек не посмотрело на исходные коды, безопасность от этого не увеличится, если смотрящие не обладают нужной квалификацией. Допустим, что на коды посмотрит миллиард людей, не понимающих в программировании, безопасность изменится? Нет. Количество реальных специалистов в безопасности, смотревших код, не так высоко, а именно они способны быстро и качественно находить ошибки. Это — отдельный навык, который требует специального обучения.

Открытость в отношении Linux имеет одно преимущество — отличное соотношение цены и качества. Возможность бесплатно установить ОС позволяет сэкономить большие деньги на установке, но увеличиваются затраты на поддержку, которая для Linux стоит достаточно дорого. К тому же администрирование Linux требует бóльших навыков и умений, чем настройка Windows. Отсутствуют мастера, которые облегчают жизнь. Необходимо знать команды Linux и уметь ими пользоваться без подсказки. Поэтому со своевременными обновлениями и свежей информацией могут возникнуть проблемы.

Из-за дорогой поддержки в конечном итоге цена владения Linux может оказаться выше, чем Windows. В Северной Америке, где оплата труда сотрудников отделов ИТ достаточно высока, Linux часто проигрывает Windows в стоимости владения.

Почему же Linux так сложен? 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 команду. Проблема только в том, что нужно помнить множество команд, но если вы их помните, то сможете выполнить требуемую операцию быстрее, чем щелчками мышью.

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 использовании 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 зависит прежде всего от человека, который ее обслуживает.

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

1.3. Ядро

Ядро — это сердце ОС, в котором реализовано управление памятью и другими ресурсами компьютера. Помимо этого оно позволяет получить доступ к различному железу. Например, ранние версии ядра обеспечивали работу только двух USB-устройств: клавиатура и мышь. Современное ядро (на момент написания этих строк ядро под номером 2.6.31) поддерживает большинство современных устройств, а дистрибутивы включают драйверы для большинства популярного оборудования.

Номер версии ядра Linux состоит из трех чисел (последнее указывается не всегда):

- первое число — старший номер, который указывает на значительные изменения в ядре;
- второе — младший номер, увеличение которого указывает на 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 число — сборка, то есть номер очередного рабочего релиза. В некоторых случаях это число опускают, ибо оно несет не такую значительную смысловую нагрузку, как предыдущие номера. Например, часто говорят о версии 2.6, и в данном случае не указана именно сборка.

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

1.4. Дистрибутивы

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

Даже если установочные версии имеют разных производителей, в качестве графической оболочки почти везде используется KDE или/и GNOME, а при отсутствии в поставке их всегда можно установить. Таким образом, вне зависимости от основного дистрибутива у всех будет одинаковый графический интерфейс.

Я долго не мог решить, какой дистрибутив использовать, но потом решил выбрать Ubuntu. Я заглянул на пару сайтов, посвященных Linux, и посмотрел в статистике (такую статистику предоставляет счетчик mail.ru) ОС, с которых заходили на сайт пользователи. Наиболее популярным оказался Ubuntu. В принципе, зная один дистрибутив, очень легко перейти на другой, ведь каждый из них — все же Linux.

Разнообразие дистрибутивов является самым слабым звеном ОС Linux (на мой взгляд). Когда вы начнете работать с ОС, то увидите, что большая часть операций не стандартизирована (это можно расценивать как следствие открытости кода). Получается как в поговорке "Кто в лес, кто по дрова". Это серьезная проблема, которая усложняет восприятие. Но в реальности в 99% случаев в дистрибутивах все идентично, поэтому проблема больше раздута, чем имеет реальную почву.

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

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/2003/2007 постоянно изменяются (только успевай привыкать к ним!). В Linux, несмотря на отсутствие стандартов, элементы интерфейса пока везде остаются одинаковыми.

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, после установки полной версии ваш компьютер сразу готов 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 сотни долларов. Поэтому реальная стоимость рабочего места на базе ОС Windows намного выше цены дистрибутива Linux.

При таком сравнении ОС Linux окажется победителем. Но, как уже говорилось, у Windows намного более дешевая поддержка, а для получения полноценной помощи в Linux нужен доступ к сети разработчика дистрибутива, который стоит достаточно дорого. Расходы на поддержку могут сделать стоимости владения этими операционными системами примерно одинаковыми. Именно поэтому я не буду вас убеждать, что Linux лучше, потому что бесплатна: это не совсем верно. Но мы увидим, что ОС Linux достаточно гибка и надежна, чтобы ее можно было выбрать в качестве рабочей лошадки для вашего компьютера. 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 большинство программ, сервисов и графическая оболочка принадлежат разным разработчикам и компаниям. Какой именно продукт будет включен в состав дистрибутива, определяется производителем.

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.

1.4.1. Red Hat Linux

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

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

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.

1.4.2. Slackware

Мое знакомство с Linux начиналось именно с дистрибутива Slackware (**www.slackware.com**). Это один из самых старых и сложных для домашних пользователей дистрибутивов. 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.

1.4.3. SuSE Linux

Мне приходилось работать с разными программами от немецких производителей, но удобство работы с ними не просто хромало, а создавалось впечатление, что эти программы — безногие калеки с детства. 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-то умудрился испортить KDE или GNOME, ведь внешний вид ОС зависит от этих оболочек. И SuSE сумела не испортить эту красоту своими картинками и логотипами.

Но нельзя сказать, что программисты SuSE вообще ничего не делали. Они добавили в дистрибутив набор утилит под названием YaST, которые значительно упрощают администрирование. Я бы посоветовал SuSE только любителям и для использования на клиентских компьютерах. Тем более, что это один из платных дистрибутивов, который распространяется в коробке.

В настоящее время этот дистрибутив находится под патронажем компании Novell.

1.4.4. 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.

1.4.5. Ubuntu

Это, наверное, один из самых простых дистрибутивов. Его основной целью во время создания была простота, а в прошлом году компания заявила, что в течение двух лет планирует сделать дистрибутив красивее и удобнее Mac OS. Пока быстрых продвижений в сторону красоты не заметно, но с точки зрения простоты дистрибутив уже давно является одним из лучших. А судя по статистике счетчиков Linux ресурсов в Интернете, в России этот дистрибутив самый популярный. Сайт разработчиков — **www.ubuntu.com**.

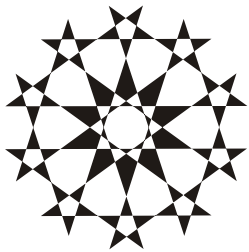
Дистрибутив построен на базе технологий Red Hat и схож с Fedora. Единственное, что меня пугает — большинство статистических обзоров показывает,

что ежегодно в этом дистрибутиве находят больше уязвимостей, чем в конкурентах. Статистикам и аналитикам верить очень сложно, особенно с точки зрения безопасности, потому что безопасность измерить невозможно. В дистрибутиве может быть найдена сотня уязвимостей, но он останется надежным, если все они незначительные. С другой стороны, достаточно одной уязвимости, но очень серьезной, которую можно легко использовать и которую залатают с большим опозданием, чтобы надежность опустилась до нуля.

Существует еще множество дистрибутивов, варьирующихся от больших и мощных систем, включающих все необходимое, до маленьких, 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 из-за большого разнообразия возможных используемых программ. Поэтому мы будем обращать внимание в основном на общие закономерности, и лишь изредка говорить об особенностях дистрибутивов.

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

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

Во время инсталляции любая ОС должна определить установленное оборудование и подготовить все необходимое для его нормальной работы. Еще лет семь назад перечень поддерживаемых устройств можно было просмотреть за несколько минут, так как 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. А базы данных драйверов в дистрибутиве Linux скоро не будут уступать Windows, а может быть, уже не уступают.

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. А может быть, что-то просто покажется веселым.

Основные принципы безопасности и производительности закладываются 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. 1.4* мы рассмотрели наиболее популярные на данный момент дистрибутивы и их основные отличия, что облегчит вам принятие правильного решения.

Лично я предпочитаю Ubuntu и Fedora. Не знаю почему и объяснить не могу, но, судя по статистике различных сайтов в Интернете, в том числе и по ОС Linux, именно Ubuntu является самым популярным дистрибутивом в России. Если устанавливать серверную версию этого дистрибутива, то он будет ставиться в текстовом режиме. Серверу не нужен графический интерфейс, он там даже лишний. Мы же в этой книге будем немного затрагивать графический интерфейс, хотя бóльшую часть времени будем разговаривать об утилитах командной строки и конфигурационных файлах. Мы будем больше говорить о серверной роли ОС Linux.

Важное замечание, которое я должен сделать, — устанавливайте самый свежий стабильный 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 еще никто не знает :). Кроме того, надо помнить, что программные средства необходимо своевременно обновлять. Если воспользоваться старым дистрибутивом, то объем обновлений может оказаться слишком большим. Не лучше ли установить сразу все новое и максимально быстро запустить сервер в эксплуатацию?

Если установить старый дистрибутив и не обновить его до последней версии, то в нем, скорей всего, будут известные хакерам ошибки, а значит, они без проблем смогут взломать сервер. Если этого не сделают сразу, то через какое-то время обязательно сделают. Если вы считаете, что ваш сервер никому не нужен, то сильно ошибаетесь. Даже пустышка кому-то нужна.

Полгода назад мой знакомый спросил меня: "А можно узнать, как взломали Linux сервер?" Конечно же можно, если журналы сохранились и их никто не успел уничтожить. Меня связали с горе-администратором, сервер которого

взломали. Первый вопрос, который я задал, оказался весьма удачным — какой дистрибутив установлен на сервере. Администратор точно не знал, потому что ему кто-то посоветовал срочно переустановить Linux. Переустановка — банальное решение, но она не всегда решает проблему, особенно, если проблема в конфигурации, а не в самом сервере.

В данном случае администратору переустановка помогла. Он нашел диск, с которого он устанавливал Linux много лет назад (и не обновлял с тех пор), и на нем было написано три циферки 6.22. Да, это очень старая версия, начала 2000-х годов. Точные годы ее появления не помню, но это было даже раньше Windows XP. В этой истории удивляет только то, как сервер смог прожить столько времени до 2008 года, когда это происходило, невзломанным. Сервер долгое время был никому не нужным, но кто-то нашел его и использовал для перекачки нелегала, и горе-администратор получил несколько гигабайт лишнего трафика, за которые пришлось платить.

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

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

Если у вас уже установлена Windows, и вы хотите, чтобы на компьютере было сразу две ОС, то придется сделать несколько телодвижений. Для установки Linux нужно пустое пространство на диске. Нет, это не свободное место на логическом диске C:, а пустота на винчестере, место, не занятое какими-либо разделами. В последних версиях инсталляторов есть возможность изменять размер раздела прямо в программе установки, причем без потери данных. Раньше для изменения разметки диска приходилось уничтожать информацию.

Если у вас все пространство диска уже используется и вы хотите откусить небольшой кусок пространства от существующего диска, то это вполне реально. Данные на существующем диске просто сдвигаются, а в опустошенной части диска может быть создан новый раздел для Linux.

Для повышения надежности процесса сдвига многие рекомендуют сначала произвести дефрагментацию. Эта операция более безопасна и заключается

в том, что разбросанные по всей поверхности диска данные собираются в одном месте. Данные одного файла могут быть разбиты на множество кусков, и лежать в любом месте диска. После дефрагментации все файлы будут занимать лишь одну непрерывную область на диске. Таким образом, при сокращении размера не придется опустошать от данных содержимое раздела, подлежащее обрезанию.

Прежде чем вставить компакт- или DVD-диск и начать установку Linux, желательно определиться с другими системами, а именно, будет ли на компьютере установлен Windows. Если да, то я бы порекомендовал сначала установить Windows, и только потом Linux. Дело в том, что установщик от Microsoft стирает загрузочную запись и уничтожает любое присутствие сторонних систем. Это значит, что установленный ранее Linux станет недоступен.

Установщики Linux более доброжелательны к чужим разработкам, и если на компьютере были "окна" от Microsoft, то загрузчик Linux позволит загружать их без проблем. Да, загрузчик Linux можно восстановить, даже если он был уничтожен установщиком Windows. Достаточно просто иметь загрузочный диск, с которого нужно загрузить Linux и выполнить команду:

```
grub-install /dev/hda
```

Но о командах мы еще будем говорить много, и это небольшое забегание вперед.

2.2. Начало установки

Итак, свободное место на диске у нас уже есть. Теперь можно приступить к запуску инсталлятора. Для этого вставьте диск в дисковод и перезагрузите компьютер. Если в BIOS (Basic Input/Output System, базовая система ввода/вывода) отключена загрузка с CD-ROM, то самое время включить ее, и тогда при старте компьютера автоматически запустится процедура установки с CD- или DVD-диска.

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**.

Мы не будем расписывать абсолютно все шаги установки, потому что большинство из них вполне понятны (выбор типа мыши, клавиатуры, способа смены языка и т. д.), и к тому же могут отличаться в зависимости от дистрибутива. Вместо этого мы опишем критические моменты установки, когда неправильный выбор может повлечь проблемы с безопасностью или эффективностью системы. И обычно первый из таких моментов — это распределение дискового пространства.

2.3. Разбивка диска

Программы установки могут поддерживать три варианта использования дискового пространства для размещения ОС:

- **Использовать весь диск** — все существующие разделы будут уничтожены, а значит, вся информация будет потеряна (рис. 2.1). Этот вариант удобен, если вы устанавливаете единственную ОС на новый компьютер. Программа установки сама выберет, сколько места и для чего отвести;
- **Использовать свободное место** — если на компьютере уже установлена 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;
- **Указать разделы вручную** — этот вариант дает возможность самостоятельно выбрать параметры создаваемых дисков. Он наиболее сложен, но позволяет добиться максимально эффективных и безопасных результатов.

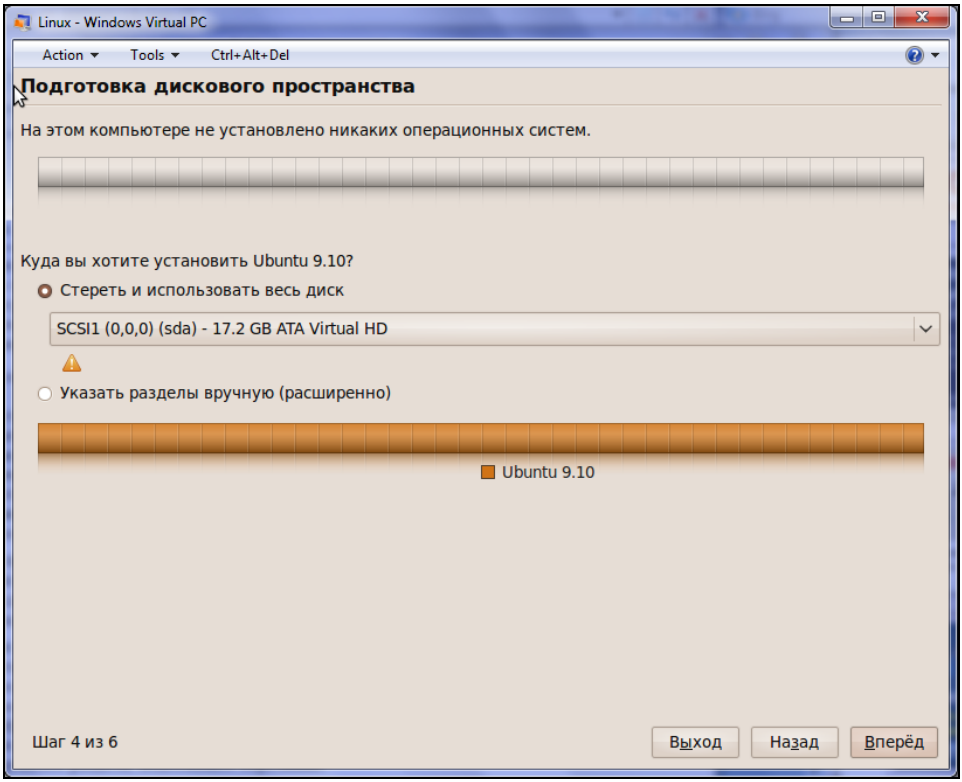


Рис. 2.1. Разрешаем системе самостоятельно разбить диск

Надо иметь в виду, что названия этих пунктов условные, так как разные разработчики именуют их по-разному. К тому же, не факт, что ваш дистрибутив поддерживает все три варианта. Возможно, присутствуют только два из них, как в моем Ubuntu (рис. 2.1), особенно, если диск совершенно пустой.

Если выбрать пункт **Указать разделы вручную**, то вы сможете самостоятельно создать разделы для Linux (рис. 2.2). В *разд. 2.3.3* мы поговорим о том, какие разделы нужны для работы Linux. В моем случае установка идет на пустой диск виртуальной машины, поэтому на нем нет разделов. Если же на вашем компьютере уже есть ОС, то на этом шаге вы можете подкорректировать диск, выделить пустое место из существующего диска или разбить диск так, как вам необходимо.

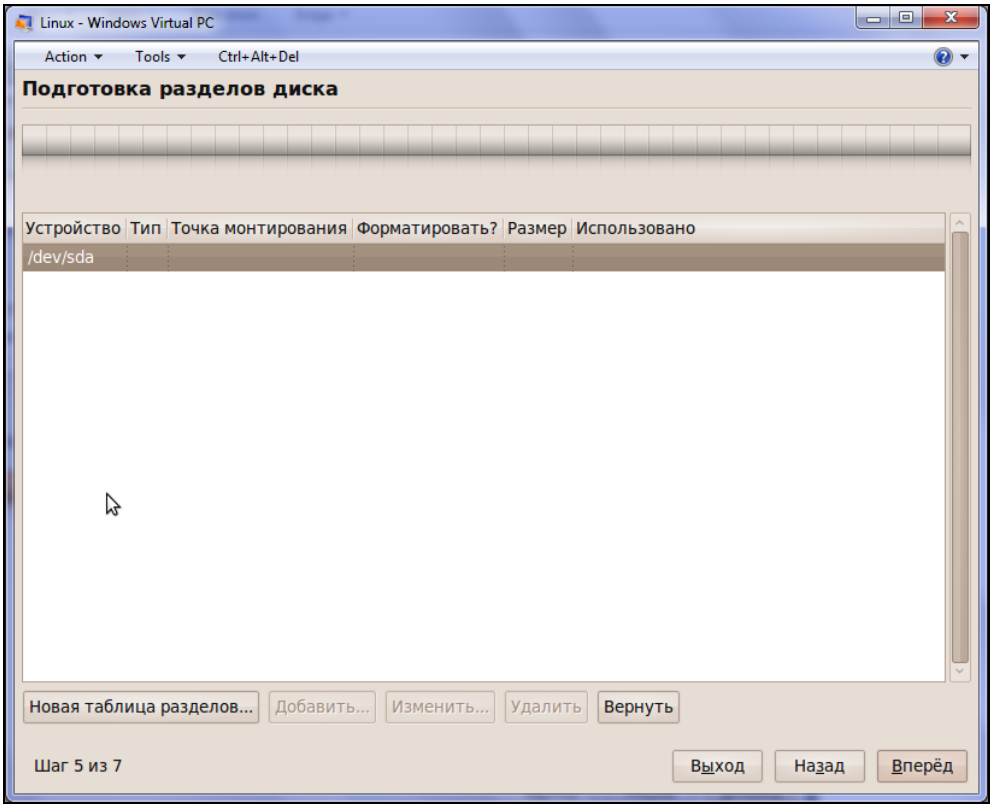


Рис. 2.2. Окно ручной настройки разделов диска

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/hdNX для дисков IDE и /dev/sdNX для дисков SCSI. В обоих случаях буква N — это номер диска. Например, если у вас два жестких диска типа IDE, то в системе они будут именоваться /dev/hda и /dev/hdb.

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

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

2.3.2. Файловые системы

Теперь поговорим о файловых системах, с которыми работает Linux. От файловой системы зависит качество хранения информации на жестком диске. Эта ОС поддерживает множество систем, в том числе и используемые 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 по сравнению с Ext2, поскольку 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 чего достигается высокая производительность. Но если возникнут проблемы с питанием или произойдет аварийный выход из системы, то компьютер может не успеть сохранить данные. При следующей загрузке ОС обнаружит нарушение целостности жесткого диска, и запустится программа сканирования диска fsck (аналог scandisk в Windows), которая восстановит его работоспособность. Однако 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, после чего проверяется целостность данных и, если данные записаны верно, 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 записать файл размером 100 байт, то блок будет уже занят, но в нем останется 90% пустого пространства, которое нельзя использовать, то есть происходит утечка памяти на жестком диске. Таким образом, на нем будет храниться немного меньше информации, чем вы ожидали. Файловая система RaiserFS позволяет заполнять блоки более полно.

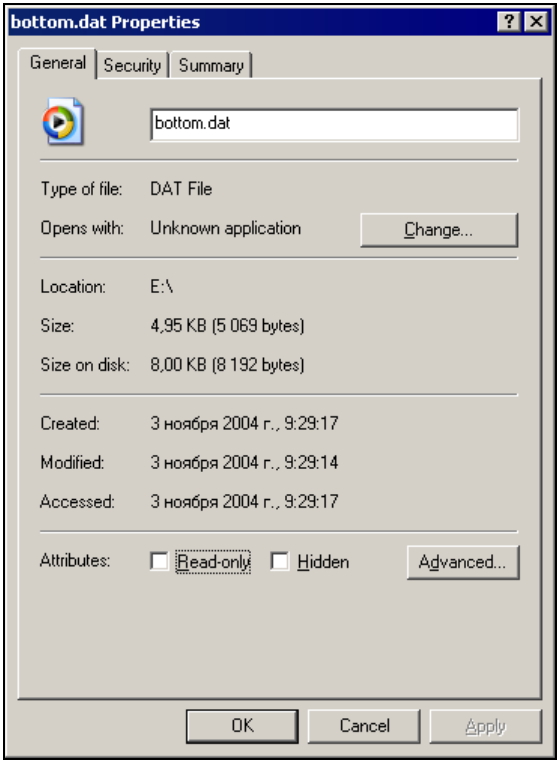


Рис. 2.3. Окно свойств файла в Windows

Утечку наглядно можно увидеть, если в ОС Windows открыть окно **File Properties** (Свойства файла), показанное на 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 килобайтам. Размер файла больше этого значения, поэтому ОС пришлось выделить два кластера, и второй заполнен менее чем на 25%. Остальное дисковое пространство пропало и не может использоваться, по крайней мере пока файл не будет стерт.

Если на диск поместить 1000 файлов по 100 байт при размере кластера 4 Кбайт, то каждый из них будет записан в свой блок. При этом на диске будет израсходовано 4 Мбайт вместо положенных 100 Кбайт. Потери пространства составят 97,5%.

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 также принадлежит к новому поколению журналируемых систем и работает аналогично ReiserFS. На данный момент она является системой по умолчанию в большинстве современных дистрибутивов Linux. Трудно сравнить по производительности ReiserFS и Ext3, но с точки зрения надежности советую использовать вторую. Разные специалисты придерживаются разных мнений, но я думаю, что стоит согласиться с мнением разработчиков и выбрать Ext3.

2.3.3. Ручное создание разделов

Если вы настраиваете сервер, а не домашний компьютер, то можно подумать о том, чтобы создать разделы вручную. По умолчанию программа установки создаст только два раздела — основной и для файла подкачки (будет использоваться при нехватке оперативной памяти), что неэффективно и даже небезопасно.

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

Таблица 2.1 *(окончание)*

Раздел	Описание
/dev	Представления для подключенных устройств
/etc	Конфигурационные системные файлы
/home	Пользовательские папки и файлы
/lib	Библиотеки ядра ОС
/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 последнего, начинаются со слэша. Это не случайно: раздел подкачки создается не как классический раздел и не становится папкой, в которой находятся файлы. Это отдельный раздел, не привязанный к корню файловой системы, который система может использовать как дополнительную оперативную память при нехватке физической, как Windows использует файл подкачки. Этот раздел не имеет смысла для пользователя, поэтому существует отдельно.

Размер раздела Swap 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 установленный объем ОЗУ, потому что память может быть расширена, и чтобы не иметь проблем с разделом подкачки, лучше заранее иметь достаточный запас. Конечно, Linux сможет работать и вообще без раздела подкачки, но все же желательно его создать. Неужели вам жалко нескольких гигабайт во имя процветания пользовательских программ?

Для работы Linux имеет смысл создать как минимум два раздела: основной (обозначается как /) и подкачки (Swap). Мы уже знаем, что второй не обязателен, но все же создайте его. Первый раздел может иметь любой тип файловой системы, кроме Swap, а второй, 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 подкаталогов. В принципе, они в любом случае будут подкаталогами к корню, просто могут располагаться каждый на своем диске для повышения надежности или производительности.

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 имеет смысл вынести папки /home и /var. Логические диски/разделы в этом случае не дадут желаемого результата.

Какими должны быть размеры разделов? Размер Swap нужно задавать в зависимости от установленной памяти, и об этом мы уже говорили. 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 вернемся. Некоторые специалисты по безопасности рекомендуют располагать этот раздел на самом большом диске (чаще всего там же содержится и раздел /home), но это ударит по производительности. Когда журналы находятся на отдельном диске, то это позволяет выполнять запись в них параллельно с обслуживанием остальных разделов. Это значит, что пользователь работает со своими файлами в разделе /home на одном жестком диске, а другой винчестер в это время сохраняет всю информацию об активности пользователя. Если обе папки будут на одном диске, то запись не сможет быть параллельной.

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 (которые достигают большого объема!), и если пожадничать, то пользователи скоро начнут возмущаться на счет нереальности сохранения на сервере результатов очередной игры. Если возможность есть, то выделите каждому из разделов максимально большой диск и забудьте про проблемы (ну хотя бы на время :)).

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

Если вы создали новый раздел, то, вероятно, следующим этапом автоматически запустится процедура форматирования раздела. Если же установка идет на существующий раздел, где уже мог стоять Linux, то программа установки запросит у вас необходимость форматирования. Если раздел содержал важные данные, то вы можете сохранить их, отменив форматирование.

2.4. Выбор пакетов для установки

Выбор пакетов — достаточно интересный момент, и именно здесь обычно допускают первую и самую страшную ошибку начинающие пользователи Linux — указывают все, что знают и, тем более, что не знают (на всякий случай). Да, название и назначение многих пакетов непонятны и незнакомы большей части пользователей, поэтому начинающие не могут четко определить список того, что им нужно. 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, проверяю работоспособность отдельных модулей, тестирую конфигурации и изучаю новые серверные программы/утилиты. Но в "боевых" системах я 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 их 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.

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

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.

Для упрощения установки некоторые дистрибутивы предоставляют предопределенные варианты установки. Они могут иметь такие или схожие названия:

- **Быстрая** — использовать определенную производителем конфигурацию. Таким образом вы сократите количество вопросов, на которые надо ответить во время установки, но результат будет далек от оптимального или безопасного;
- **Разработка** — задействовать основные пакеты и все необходимые компоненты для разработки приложений, компиляции ядра ОС Linux и т. д.;
- **Офис** — установить типовые пакеты плюс офисные приложения;
- **Клиент** — включить в устанавливаемую версию ОС только клиентские программы;
- **Сервер** — установить только серверные программы для выполнения определенной роли в вашей сети;
- **Выборочная** — самостоятельно выбрать требуемые компоненты. Это наиболее предпочтительный вариант для серверов и компьютеров, которые будут подключены к сети;
- **Обновление существующей версии** — сохранить многие установки системы, что позволит потратить меньше времени на повторную настройку. Разумеется, такой вариант может появиться, только если ОС Linux уже установлена.

Названия этих пунктов и их количество зависят от дистрибутива и могут очень сильно отличаться в каждой версии. Последний пользовательский

дистрибутив Ubuntu вообще ничего у меня не спросил, а сразу начал ставить заранее предопределенный разработчиками список программ.

В этот же момент иногда предлагается выбрать графическую оболочку — KDE или GNOME.

Никогда не выбирайте серверную установку. В этом случае на компьютере отсутствуют клиентские приложения, но зато будут работать в фоновом режиме всевозможные программы-демоны. Это как раз самый опасный вариант, если не отключить ничего лишнего. Если офисные программы не открывают портов, не работают с сетью (то есть 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, что еще хуже, каждый лишний демон — это удар по безопасности.

Для начинающего можно остановить свой выбор на пункте **Разработка** или **Офис**, чтобы у вас были все необходимые возможности для написания программ и работы с документами, но при этом в компьютере не устанавливались серверные приложения. К тому же, так вы сможете познакомиться к графическим режимом Linux. После определения типа инсталляции укажите пункт **Выборочная**, чаще всего я его видел внизу окна. Это позволит вам выбрать дополнительно требуемые пакеты.

На следующем этапе мы увидим список всех компонентов, которые могут быть установлены. Если заранее известно о необходимости использования какого-либо сервиса, 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 вы сможете расширить возможности. Удалить, в принципе, тоже можно, но лень побеждает все, и вы забудете это сделать, поэтому лучше не поставить что-то, чем установить лишнее. Помните, что уже на этапе установки мы закладываем фундамент будущей производительности и безопасности системы. Если вы не пользуетесь какой-либо программой, то, конечно же, не будете следить за ее обновлениями и не станете заниматься исправлениями ошибок. А злоумышленник для повышения своих привилегий может ей и воспользоваться. Таким образом, вы откроете хакеру дверь в вашу систему нараспашку.

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, особенно, если вы используете старую машину.

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 задачи по разным компьютерам, да 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, о которых пока еще никто не знает, 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, с легкостью получить доступ к FTP-серверу и скачать все секретные данные. Если на взломанном 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 на FTP- или новостном сервере.

Пример с WEB-сервером не вполне нагляден, потому что этот сервис умеет делать многое. Если хакер сможет закачивать свои файлы на сервер, то он сможет загрузить для себя WEB Shell и выполнять команды, закачивать/скачивать через WEB. Да и FTP-сервис чаще всего нужен пользователям именно на WEB-сервере, а не на каком-то другом компьютере.

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. Это неправильно. О паролях мы поговорим в *разд. 2.6* и *главе 3*, но уже сейчас вы должны знать, что для каждой системы должен быть свой код доступа, а пароли сетевых сервисов должны быть максимально сложными.

Но не только демоны являются потенциальной проблемой. В состав Linux многие программы включаются в исходных кодах и должны компилироваться

в машинные перед выполнением. Программы, использующие уязвимости Linux-систем, также часто поставляются в исходниках. 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. Это мелочь, но из мелочей строится вся наша жизнь.

В ОС Linux очень редко используются инсталляторы программ, поэтому все настройки производятся во время компиляции исходного кода. Если gcc будет недоступен, то у взломщика возникнут проблемы с выполнением зловредного кода. Его нужно будет скомпилировать заранее, и только потом закачать. Не у каждого хакера есть такая возможность, потому что большинство — это скрипткидди, которые сидят в Windows :).

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!

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.

В Linux существует множество загрузчиков, а самыми популярными являются LILO (Linux Loader) и GRUB (Grand Unified Bootloader). Первый из них уже давно не используется большинством, и сейчас, кажется, уже все используют GRUB, поэтому выбора практически нет. 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).

Двигаемся дальше. При наличии сетевой карты в компьютере вы увидите окно для ее выбора. Если нужный драйвер в списке не будет найден, то можно оставить значение **none**. Это не значит, что сетевая карта не будет работать, просто будет использоваться универсальный драйвер. В дальнейшем я реко-

мендую установить корректные драйверы, чтобы заставить сетевую карту работать на все 100%. Некоторые дистрибутивы этот шаг пропускают, потому что Plug&Play позволяет определить тип карты достаточно точно и выбрать наилучший драйвер из большой базы, которая уже есть на диске.

Теперь поговорим о настройке сети. Если в вашей сети используется единственный DHCP-сервер, то можно оставить все по умолчанию. Если же 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 192.168.1.1. 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. Этой информации хватит для начальной настройки.

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, под которым будете работать, и его пароль, и в зависимости от ваших прав вам будет предоставляться доступ к определенным разделам и функциям ОС.

В дистрибутиве может быть возможность выбрать учетную запись, под которой будет осуществляться автоматический вход в систему при загрузке компьютера. В погоне за простотой кто-то придумал такую фишку, чтобы Linux был похож на Windows 95 или Windows XP Home и пользователь не думал о пароле, но я бы не советовал пользоваться этой возможностью. Не ленитесь вводить пароли.

Программа установки моего дистрибутива 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.

Проверка длины пароля — это уже хорошо, ведь это значит, что у администратора просто обязан быть пароль и пустой вариант будет невозможен. Все специалисты по компьютерной безопасности в один голос просят пользователей не задавать простые пароли, но мало кто следует этим рекомендациям. Нельзя для этих целей использовать имена, читаемые слова или даты рождения.

Такие пароли легко взламываются простым перебором по словарю, и это не отнимет много времени, если словарь хорошо составлен. А если систему будет взламывать человек, который вас знает, то он сможет обойтись и без словаря, а завершит ручной подбор за несколько минут.

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, дефис или подчеркивание). Длина пароля должна быть не менее 8 символов, а лучше — более 12. Тогда для подбора хакеру потребуется очень много времени. Не один здравомыслящий человек на такое не согласится.

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, хотя 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, а 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 сразу очень существенно усложнится. 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 пароля администратора, вам может быть предложено 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 и только при крайней надобности переключаться на учетную запись root, чтобы выполнить административную задачу.

Если программа установки вашего дистрибутива предложит завести непривилегированного пользователя (последние дистрибутивы, кажется, все уже

делают такое предложение), обязательно сделайте это и впоследствии работайте именно под этой учетной записью, а если для выполнения определенной команды не хватает прав, переключайтесь на учетную запись администратора. Если такого предложения не поступит, то после перезагрузки создайте простую учетную запись самостоятельно.

Когда установка будет завершена, можно перезагружать компьютер, вытащив CD-диск из привода.

2.7. Первый старт

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

Загрузчик системы может скрывать процесс старта, но где-то под графической оболочкой обязательно скрывается запуск сервисов. В старых системах этот процесс отображался в виде текстовых строк, которые бежали по экрану.

ОС Linux — многопользовательская. Это значит, что с ней могут работать несколько человек. Система должна знать, с кем она сейчас работает, поэтому после загрузки ОС вам предложат представиться, указав имя пользователя и пароль. Первый параметр позволяет идентифицировать вас, а второй — обезопасить от нежелательного использования вашего имени другим пользователем. Идентификация пользователя в текстовом режиме выглядит как строка с приглашением ввести имя:

```
localhost login:
```

После ввода имени пользователя вы должны указать пароль, чтобы ОС смогла определить, что вы являетесь тем, за кого себя выдаете. Если вы выбрали во время загрузки графический режим и правильно настроили видеокарту, то перед вами появится графическое приглашение входа в систему. Современные дистрибутивы устанавливают графический вход по умолчанию (рис. 2.4), поэтому текстовое приглашение можно увидеть при неправильной настройке видео или при установке серверной версии Linux без установки графических оболочек.

Прежде чем вводить имя, осмотритесь. Возможно, где-то есть какие-то кнопки или меню. Например, у Mandriva 2008 внизу есть кнопочка **Тип сеанса**, нажатие которой позволяет сменить графическую оболочку, а оболочек для Linux множество. Самые распространенные из них — это GNOME и KDE. Наиболее популярный дистрибутив Ubuntu устанавливает только GNOME. Если вы предпочитаете KDE, то следует устанавливать Kubuntu, который выпускается тем же производителем, а именно Canonical Ltd.

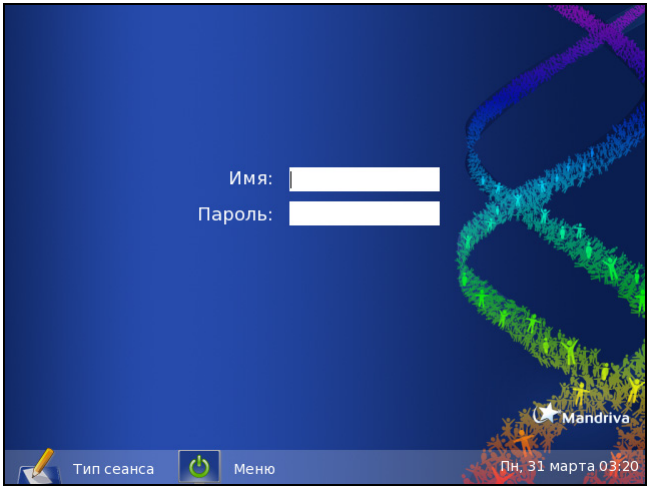


Рис. 2.4. Вход в Linux Mandriva

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, что угодно. Эта власть нередко приводила к плачевным последствиям, когда администраторы во время тестирования каких-либо параметров по ошибке уничтожали важные данные.

И дабы защитить нас, в современных системах по умолчанию уже запрещено входить под именем root в графическом режиме. Да, это ограничение можно обойти, и возможно, что обход будет доступен и в будущем. Однако если раньше нас только предупреждали об опасности работы под root, то теперь запрещают вход.

Работа под учетной записью root может облегчить взлом компьютера через простые на первый взгляд приложения и способствует вирусным эпидемиям. До сих пор в UNIX-подобных системах почти не было вирусных эпидемий как раз из-за того, что пользователи системы не обладают полными правами. Допустим, что вы путешествуете по WEB-сайтам. Запущенный вами 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 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.

> ***Примечание***
>
> Необходимо заметить, что некоторые команды, которые мы будем описывать в книге, могут оказаться недоступными от имени простого пользователя. Так как мы рассматриваем настройки системы, то некоторые действия будут опасны, поэтому доступны только администратору. Если при попытке выполнить что-либо произошла ошибка, например "команда не найдена", попробуйте переключиться на root, и, скорее всего, это поможет. Хотя, возможно, и нет, если у вас экзотическая версия или вы не установили нужную программу.

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 имя пользователя, которое может быть различным в каждом терминале.

Использование терминалов — очень удобная возможность. Например, 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 же вы выбрали текстовый режим, то в любой момент можно переключиться в графический, набрав команду `startx`. В этом случае графическая оболочка будет загружена автоматически, без приглашения ввести пароль, потому что вы уже идентифицировали себя в текстовой консоли.

Текстовое приглашение на вход в систему может появиться и в случае ошибки конфигурации. При этом выполнение команды `startx` ни к чему не приведет, так как графическая оболочка не сможет загрузиться. Тогда придется произвести настройку графики заново. Бóльшая часть 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 настройки системы. Выберите пункт **X Configuration**. Система скорей всего сама определит наличие видеокарты и необходимые драйверы, а вот с монитором могут быть проблемы. В старых дистрибутивах 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 работоспособность выбранного графического режима. Если все пройдет удачно, то программа предложит использовать графический режим для входа в систему.

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

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.

Познакомимся с внешним видом Рабочего стола в графических оболочках Linux. На рис. 2.5 показано окно графической оболочки GNOME. В нижней или верхней части Рабочего стола обязательно 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;
- кнопки быстрого вызова основных программ — чаще всего это несколько ярлычков для таких программ как браузер, окно терминала, текстовый редактор и еще что-нибудь на выбор разработчика;

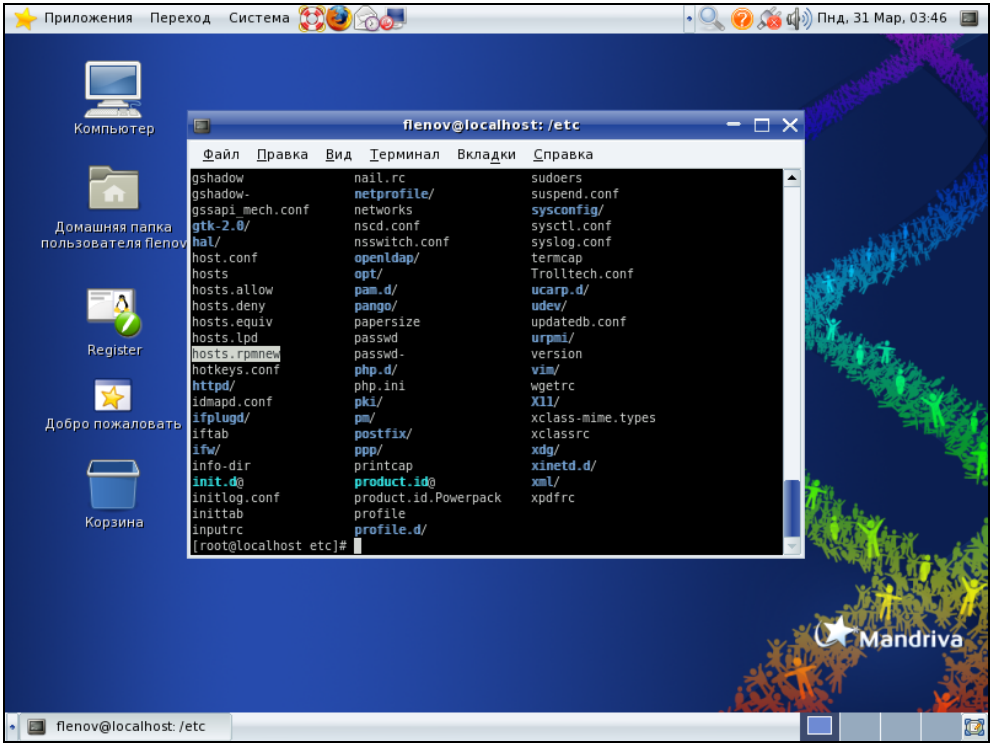


Рис. 2.5. Рабочий стол в графической оболочке GNOME

- пустое поле — здесь будут располагаться кнопки запущенных программ. Кнопки задач можно использовать для переключения между работающими приложениями;
- часики, без которых жизнь просто невозможна. Куда же без этого необходимого атрибута любого рабочего стола.

Теперь самое время познакомиться с окном терминала, в котором можно выполнять команды 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.

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. Таким образом, вы всегда знаете, где находитесь, и где будут по умолчанию выполняться вводимые команды.

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 используется каталог /home/root. Если вы входите в систему под именем пользователя michael, то вашей домашней папкой будет /home/michael. В этой папке вы можете хранить свои файлы. Домашнюю папку пользователя root лучше не трогать.

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

В Linux существует несколько командных оболочек, и каждая из них наделена своими возможностями. Чаще всего пользователи применяют оболочку под названием bash (Bourne Again Shell). Мы будем рассматривать именно ее, как наиболее популярную, и именно ее назначает пользователям Ubuntu. Другие оболочки похожи, но немного отличаются возможностями. Когда вы запускаете терминал, то текущей становится ваша домашняя папка.

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 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.

Да, сейчас ОС Linux обзавелась графическим интерфейсом и если вы используете его, то выключение компьютера не вызовет проблем. Но если вы используете текстовый режим, то без команды корректно выключить компьютер проблематично.

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`.

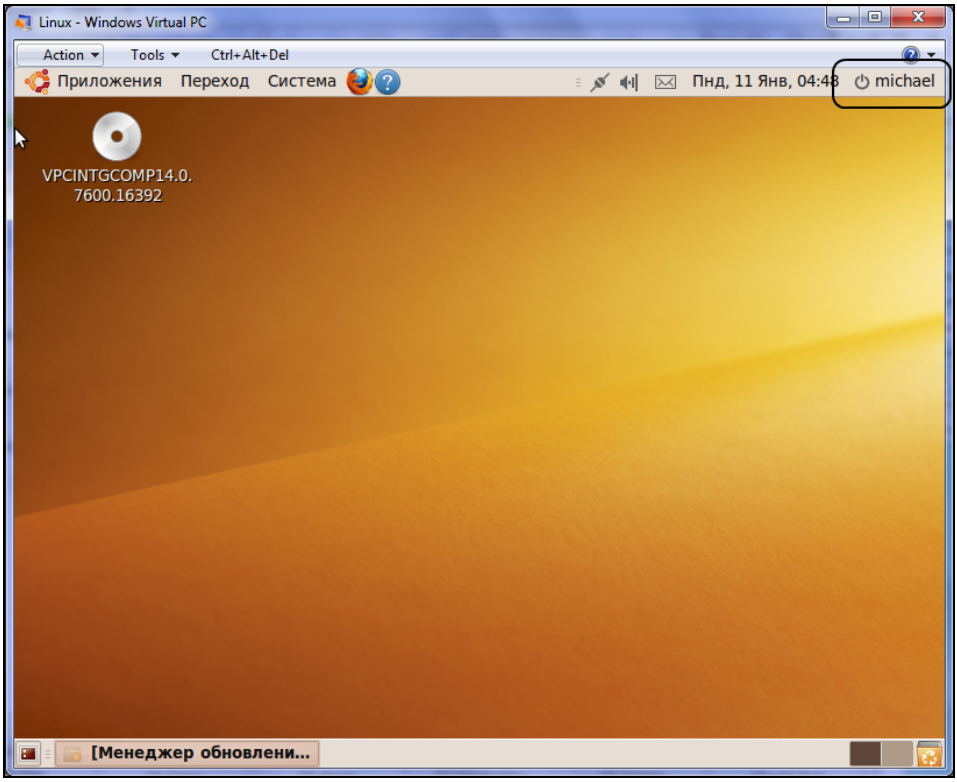


Рис. 2.6. Завершение сеанса в оболочке GNOME

Чтобы выйти из графического режима, нажмите кнопку вызова главного меню и выберите пункт **Выйти из системы**. Это заставит графическую оболочку выгрузиться, после чего появится окно-приглашение ввести имя пользователя, в котором можно войти в систему под новым именем или перезагрузить компьютер.

В графической оболочке GNOME нужно выбрать меню **Система | Завершить сеанс...**. В последней версии кнопка выключения компьютера из графического режима находится в главном меню справа от часиков (рис. 2.6).

2.9. Подсказки

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

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

Для получения более подробной информации нужно воспользоваться следующей конструкцией:

```
man имя
```

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

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

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

Если вы устанавливаете лицензионный вариант ОС, то обычно в коробке можно найти документацию по установке и руководство пользователя. Чаще всего эта информация поверхностна и позволяет сделать только минимальные настройки. Но некоторые дистрибутивы содержат в поставке 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 типа ОС и конкретного сервиса. Придерживаясь их, вы 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 можно выразить следующими двумя тезисами.

1. Необходимо запускать минимум возможного. Это касается не только сервисов в целом, но и их составляющих. Например, если вам нужен WEB-сервер, 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 них, 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. Одновременное использование обеих технологий — абсолютно бесполезная и глупая затея.
2. Следует разрешать минимум необходимого. Многие администраторы не любят заниматься какими-либо настройками, поэтому открывают полный доступ ко всему, 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 часто открывают какую-то папку для всеобщего доступа. Это мотивируется тем, что пользователям может потребоваться обмен данными. А если нет?

Рассматривая конфигурирование ОС и ее сервисов, я буду часто напоминать об этом правиле, и при разборе примеров мы 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 и могли приступить к работе сразу после установки программы.

Это значит, что будет разрешено абсолютно все, а это уже нарушает первое рассмотренное правило. Даже если установка по умолчанию не делает доступными все компоненты программы, чаще всего настройки все равно оказываются небезопасными, потому что могут содержать опасные параметры или пароли по умолчанию.

Если используется не так много команд и параметров, то лучше заняться конфигурацией программы с чистого листа. Если процедура достаточно 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 (Redhat Package Manager), которые не общаются с пользователем для задания параметров установки. При этом пароли чаще всего даже не предлагается сменить и при первом старте. 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. Это имя 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, иначе все попытки защитить компьютер могут оказаться бессмысленными.

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

После того как система несколько раз оказывалась уязвимой, я взял за принцип разрабатывать программу под своей учетной записью и с измененным паролем. Лучше изменить пароль 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 хакеру достаточно увидеть устройство своими глазами, чтобы узнать пароль.

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

Я думаю, что не нужно напоминать о необходимости задания сложных паролей. На моей нынешней работе администратор баз данных выдумывает "сложнейшие пароли" — идентичные учетным записям. Как много понадобится времени на подбор пароля?

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 компании 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 и 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, забив журнал мусором до пределов диска или съев все ресурсы компьютера, заставив его только и делать, что писать в журнал. От переполнения диска защититься можно, если реализовать цикличный журнал, но в этом случае можно реализовать атаку, при которой хакер будет затирать данные журнала.

Различных вариантов нападения очень много, и защищающаяся сторона всегда находится в немного более уязвимой позиции. Нужно выбирать золотую середину.

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, умноженным 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. У вас всегда должен быть под рукой большой запас мощности, ибо иногда превышаются даже самые пессимистические оценки нагрузки.

2.10.6. Внимательность

Одной из серьезнейших угроз для компьютерной системы является внимательность, а точнее, ее отсутствие. Отсутствие внимательности присуще многим, и мне в том числе, и я, работая над книгой, регулярно что-то упускаю из виду и делаю ошибки. Конфигурация ОС Linux и ее сервисов в этом отношении более чувствительна, особенно если для настройки править непосредственно конфигурационные файлы. Одна маленькая ошибка может остаться незамеченной, но очень сильно повлиять на работу сервера.

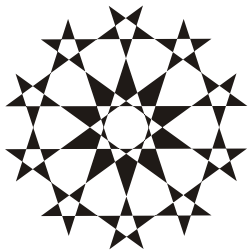
Графические утилиты более безопасны, потому что каждое действие проверяется и контролируется программой, чего не может сделать текстовый редактор. Графические программы могут быть удобнее и нагляднее (хотя и не всегда являются таковыми, потому что это зависит от их разработчика), но конфигурирование файлов напрямую более эффективно, особенно при удаленной настройке.

2.11. Обновление

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

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

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



Глава 3

Добро пожаловать в Linux

В этой главе мы начнем знакомиться с самим Linux. Надеюсь, что вы уже установили систему, а не просто прочитали предыдущую главу, потому что все, что мы будем рассматривать, лучше всего будет тут же проверять на практике. 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. 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! Лучше освободить ее 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.

Между прочим, в Windows 2008 есть сервер Windows Core, который запускается в виде окна терминала, в котором можно управлять сервером из командной строки. Создавая эту возможность, Microsoft прислушалась к администраторам, которые уже давно требуют текстового режима в этой ОС. Ну зачем нужно загружать сложные графические библиотеки простому файловому серверу? Вот и я говорю, что абсолютно не нужно.

Если вы настраиваете домашний сервер для маленькой сети, 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 ее 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, которые находятся в корне файловой системы.

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 рассказать лишь об одной программе: Midnight Commander (MC). Это лучшее средство для решения всех описанных выше задач. Программа присутствует в большинстве дистрибутивов. 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.

Файловый менеджер Midnight Commander является аналогом знаменитого файлового менеджера Norton Commander, работавшего в MS DOS. Их функции и даже внешний вид очень похожи. Если вы знакомы с той старенькой утилитой, то будет очень приятно ностальгировать по старым добрым временам, сидя в Linux. Программа может работать как в текстовом режиме Linux, так и в окне терминала оконного менеджера.

Почему я начинаю знакомство с файловой системой с этой программы? Она проста, удобна и наглядна. Не нужно запоминать множество команд команд-

ной строки, и с этой утилитой любят работать не только начинающие, но и профессионалы.

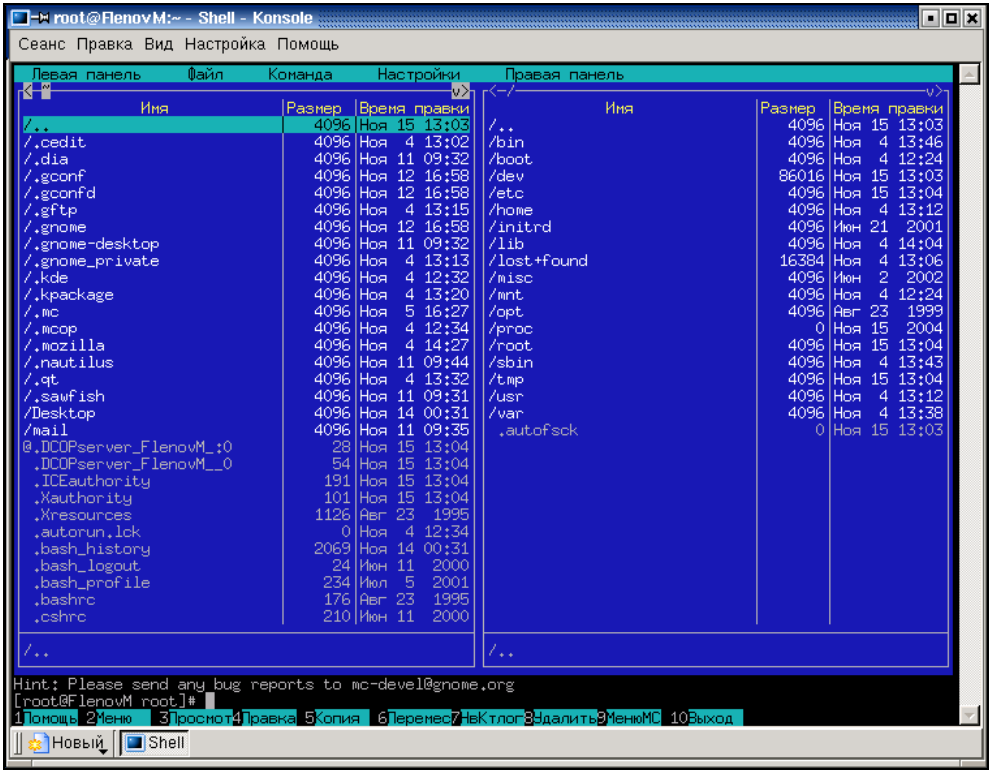


Рис. 3.1. Окно терминала с запущенной программой Midnight Commander

На рис. 3.1 показано окно терминала, в котором запущена программа MC. Окно состоит из двух независимых панелей, в каждой из которых вы можете видеть файлы и папки текущей директории (имена папок начинаются с прямой наклонной черты /). 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 из 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 /. Именно он является вершиной пирамиды в иерархии всех папок. Например, папки пользователя находятся в каталоге /home. Тогда /home/flenov будет определять путь к пользовательскому подкаталогу пользователя flenov. Чтобы попасть в эту директорию, нужно навести на нее курсор и нажать <Enter>.

В списке папок и файлов самой первой всегда стоит папка с именем, состоящим из двух точек (/..). В реальности каталога с таким именем не существует. Это указатель на родительскую директорию, с помощью которого вы можете попасть на уровень выше. Например, вы находитесь в подкаталоге /home/flenov. Если войти в папку /.., то вы подниметесь на предыдущий уровень и окажетесь в директории /home.

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

- **1 (Помощь)** — отображение 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 копирования. 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, чаще всего являются конфигурационными. Будьте осторожны при их перемещении и редактировании. Конфигурационные файлы нуждаются в максимальной защите.

Конечно, вы можете создать собственный файл с именем, у которого в самом начале стоит точка, который не будет конфигурационным, а есть множество файлов без точек, являющихся конфигурационными. Так что точка не является обязательным индикатором, но призвана привлекать внимание.

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, если вдруг заблудитесь в этом прекрасном пингвиньем лесу.

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 которых начинаются с точки) являются скрытыми. Чтобы их вывести, нужно указать ключ `-a`:

```
ls -a
```

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

```
ls -al
```

Но такая команда отобразит файлы текущего каталога, и не факт, что мы сейчас находимся, например, в каталоге /etc, который хотим просмотреть. Чтобы увидеть его содержимое, 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   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. О них мы будем подробно говорить в *главе 4*. Сейчас нам главное 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;

> ***Примечание***
>
> На первый взгляд, директория — не файл и не имеет размера, но не стоит удивляться, что размер директории не равен нулю. Грубо говоря, директории — это файлы, в которых находится список файлов директории. Размер тоже не случаен — четыре килобайта (4*1024), что равно блоку памяти (странице), выделяемому для работы с данными. Это лирическое отступление, которое может и не пригодиться в реальной жизни.

- `Nov 26 16:10` — дата и время последнего изменения файла;
- `.` — имя файла. В данном случае это ссылка на текущую директорию.

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
```

Это сработает, 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), и она 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 перейти в папку flenov, которая располагается внутри текущей, то достаточно набрать только имя папки flenov:

```
cd flenov
```

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

```
cd ..
```

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

cp

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

1. Копировать содержимое файла в другой документ той же папки:

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

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

2. Копировать файл 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. Копировать несколько файлов в новый каталог. Для этого нужно перечислить все файлы-источники и последним параметром указать папку:

   ```
   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. Копировать группу файлов каталога или все лежащие в нем файлы.

   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`.

Это далеко не полный список возможностей команды копирования. Она достаточно сложная и мощная, и позволяет выполнить за один вызов любую задачу копирования.

find

Для поиска информации в файловой системе используется команда `find`. Файловая система сильно разветвлена, и найти нужный файл бывает непросто. В простейшем варианте команда выглядит следующим образом:

```
find path -name filename
```

После имени команды идет путь к директории, в которой нужно производить поиск. Поиск будет включить все поддиректории. Вы можете указать несколько директорий подряд, например:

```
find /etc /home -name filename
```

Эта команда произведет поиск в /etc и в /home.

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

```
find / -name passwd
```

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

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

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

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

- `-size число` — позволяет указать размер файла, если вы его точно знаете. По умолчанию размер указывается в блоках размером в 512. Если нужно указать размер в байтах, то после числа укажите букву `c`;
- `-print` — заставляет вывести содержимое файла в консоль;

- `-mtime число` — в качестве числа можно указать количество дней, прошедших со дня последнего редактирования файла. Например, если вы вчера редактировали файл с именем note, но не помните, куда его сохранили, то выполните следующую команду:

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

- `-type тип` — ключ позволяет указать тип файла. В качестве типа можно указывать:
  - d — каталог;
  - f — обычный файл;
  - p — именованный канал;
  - l — символическая ссылка.

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

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

В данном примере в качестве начальной директории для поиска указана точка, то есть поиск будет начат с текущей папки, но вы можете указать другое, потому что это просто пример. Потом указываем тип искомых файлов. В качестве типа ограничиваемся только файлами, чтобы программа не пыталась мучить директории. В качестве имени указываем только звездочку, что соответствует всем файлам. Ключ `-print` заставляет отображать содержимое файла.

После этого идет самое интересное. После вертикальной линии идет другая команда. Это значит, что результат работы команды слева будет передаваться команде справа. Команда `xargs` объединяет полученные на входе данные и выполняет указанную команду. А вот команда `grep` ищет в тексте, который подается на вход, строку, указанную в качестве параметра.

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 (которая должна быть пуста):

```
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 предупреждений. Вы должны быть уверены, что команда написана правильно, иначе можно удалить что-то лишнее, особенно если вы работаете под учетной записью root.

Пример удаления директории рекурсивно и без запроса на подтверждение:

```
rm –rf /home/flenov/dir
```

df

Эта команда позволяет узнать свободное место на жестком диске или в 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` — как смонтирована файловая система.

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, по крайней мере в текстовом режиме, и многие не могут сжиться с этой особенностью. Графические оболочки прекрасно научились монтировать диски

автоматом, но все же желательно помнить и хоть немного уметь пользоваться утилитой.

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 сведений, чем просто команда `mount /dev/cdrom`. Эти сведения хранятся в двух файлах, уже имеющихся в ОС и описывающих основные устройства и параметры по умолчанию — в файлах 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 Ext3.

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. Для CD-ROM-а 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 внешних приводов и уст-

ройств, которые злоумышленник может использовать для физического переписывания информации 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
```

> ***Примечание***
>
> Mandriva 2008 при выборе разбиения дискового пространства по умолчанию создает два раздела, один основной, а на втором удобно располагается /home.

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`. Это означает, что по умолчанию диск будет смонтирован для чтения и записи (`rw`), а при возникновении ошибок перемонтирован только для чтения (`errors=remount-ro`). В качестве опций монтирования дополнительно можно использовать:

- `noexec` — запрещает выполнение 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 будем говорить не раз (например, в *разд. 4.5*), ибо это очень опасная вещь;
- `nodev` — запрещает использование файлов устройств;
- `nosymfollow` — запрещение использования 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. Нет, параметр использовать можно, но не стоит считать его панацеей.

Допустим, что ваш сайт разрабатывается с использованием языка Perl. Если его интерпретатор доступен для выполнения, то взломщик сможет запускать сценарии программ Perl в любом разделе, в том числе и с установленным параметром `noexec`. 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 этот диск принадлежит Windows (у меня именно так, поэтому

мне придется так считать). 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. В параметре указана файловая система `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
```

Вытащить смонтированный CD-ROM-диск действительно нельзя, но если была смонтирована флешка, то ее всегда можно выдернуть физически, и тут ничего не спасет.

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 развертывания 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 *главе 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, так как 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 пакета. Или запустите upgrade для обновления уже установленного пакета.

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 директории, содержащие исполняемые файлы. Например, чтобы определить, где находится программа просмотра содержимого каталогов `ls`, выполните следующую команду:

```
which ls
```

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

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

3.1.2. Безопасность файлов

В *главе 4* мы будем подробно говорить о правах доступа. Это основа обеспечения безопасности, но надеяться только на них нельзя. Необходимы дополнительные инструменты сохранения целостности системы, или, по крайней мере, отслеживание изменений 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 11:40 21 января. 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 11:40 21 января 2010 года, можно следующим образом:

```
touch 0121114010 /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 поиск.

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` — директория, в которой нужно искать. В нашем случае указана системная директория /etc, в которой 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/*
```

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

```
783fd8fc5250c439914e88d490090ae1  /etc/DIR_COLORS
e2eb98e82a51806fe310bffdd23ca851  /etc/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, поскольку 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 целью атаки 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 по своему усмотрению, добавив в него необходимые возможности. 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.

Следить за пользовательскими файлами просто не имеет смысла, потому что эти файлы изменяются часто, и в тонне изменений найти что-то важное будет проблематично. А вот конфигурационные файлы и программы в настроенной системе не изменяются и не должны изменяться, поэтому любые корректировки указывают на возможную проблему или вторжение.

Замечания по работе с файлами

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

Самое страшное — это возможность использовать невидимые символы, так как 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, в конце имени, и невнимательный администратор ничего не заметит. Чтобы увидеть конечный пробел, можно при выводе добавлять к каждому имени символ косой черты (/). 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.a11ow. Ничего не замечаете подозрительного? При беглом взгляде ничего обнаружить невозможно, но если приглядеться повнимательнее, то станет видно, 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-либо заподозрить, потому что если человек каждый день видит одно и то же, то, чаще всего, воспринимает желаемый текст за действительный. Получается банальный обман зрения.

Внимание — главное оружие администраторов. Вы должны проявлять интерес к любой мелочи, и нельзя позволить обмануть ваше 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 удобно, если копия этого файла находилась бы в домашних директориях этих

пользователей. Создавать несколько копий неудобно, потому что затруднится синхронизация изменений. 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, а его имени сопоставляется 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 проверить все, что утверждается дальше, создайте в своей системе файл 1.txt. Для этого можно выполнить команду:

```
cat > 1.txt
```

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

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

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

С помощью команды `cat link.txt` выведите на экран содержимое файла link.txt и убедитесь, что оно идентично строкам в 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` (результат отображения 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 значение счетчика ссылок в третьей колонке уменьшилось до единицы.

Символьная ссылка указывает не на данные, а на имя файла. Это дает некоторые преимущества, но одновременно вызывает 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. В третьей колонке стоит единица, а последняя колонка 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 дистрибутив Red Hat, 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", то есть 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, ведь пользователь сможет создать файл, который был удален.

Второй недостаток символьных ссылок кроется в правах доступа, но мы их будем рассматривать в *главе 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 пользователь в большинстве случаев должен иметь 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 делать это только в крайнем случае, когда все остальные способы решения проблемы еще хуже. Но если нет другого выхода, то будьте аккуратны и внимательны.

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

В ОС есть множество системных сервисов, которые работают в фоне, отнимают ресурсы при старте и во время работы, а также влияют на безопасность. Для управления сервисами в системе обязательно должна присутствовать утилита. Например, в Mandriva 2008 управление происходит из центра управления (рис. 3.2). Здесь из раздела **Система** можно запустить утилиту Manage system service by enabling or disabling them (Управление системными сервисами с помощью их включения или выключения). Может, у разработчиков не хватило сил, или они не знают, как перевести эту надпись, но в моей версии она осталась на английском.

Запустите эту утилиту, и перед вами появится окно (рис. 3.3), в котором можно запустить или остановить установленную службу, а также можно поставить галочку **При запуске**, если нужно, чтобы служба запускалась автоматически.

Если вы устанавливали какой-то 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 автозапуск 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, когда необходимо, и по окончании отладки прекращаю работу этого демона.

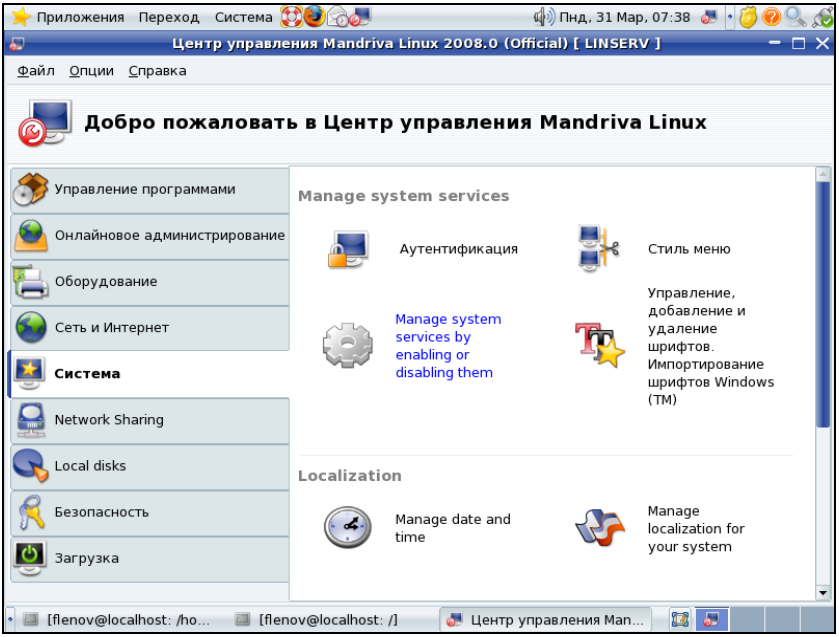


Рис. 3.2. Центр управления Mandriva Linux

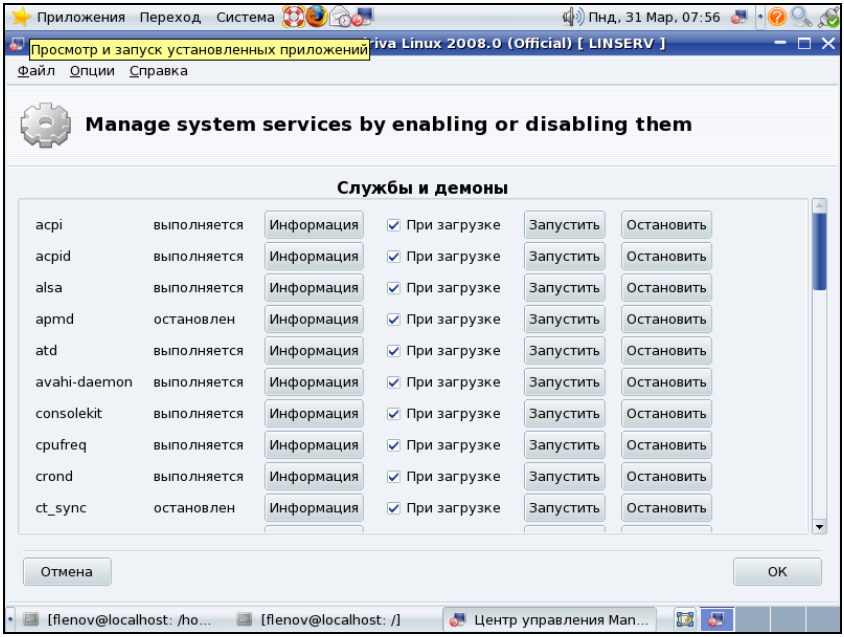


Рис. 3.3. Настройка служб

В некоторых дистрибутивах прямо на рабочем столе есть значок **Control Panel** (Панель управления), при щелчке по которому откроется окно, содержащее ссылки на основные программы конфигурирования системы. Нас будет интересовать ярлык **Службы**.

> ***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 их использовать.

3.2.2. GRUB

ОС сама по себе не загружается, когда вы включаете свой компьютер. Ее запуск должен быть как-то инициализирован. Около 10 лет назад самым популярным инициализатором этой загрузки была lilo, но за последнее время уже, кажется, все дистрибутивы перешли на GRUB (GRand Unified Bootloader, или большой объединенный загрузчик).

Когда мы включаем компьютер, то первым запускается BIOS. Его загрузка может быть скрыта красивой картинкой с логотипом производителя, но раньше в этот момент мы обязательно видели логотип БИОСа и бежали циферки тестирования оперативной памяти.

После загрузки БИОСа должен запуститься загрузчик, который и будет отвечать за загрузку ОС. Если вы установили Linux, то в этом месте у вас скорей всего запустится GRUB. Эта программа загрузчика умеет загружать не только Linux. Если у вас на компьютере установлено несколько систем, появится меню выбора, какую именно систему вы хотите загружать. В качестве нескольких систем может выступать как несколько установленных различных дистрибутивов Linux, так и Windows. Если вы выберете в меню загрузку Windows, то GRUB передаст управление загрузчику Windows, а если выберете Linux, то начнется загрузка этой системы.

Мощность GRUB заключается в том, что он поддерживает различные файловые системы и позволяет даже работать с файлами без загрузки ОС. Например, для просмотра содержимого файла можно использовать команду `cat`, которая идентична одноименной команде Linux.

Все, что касается программы загрузчика GRUB, находится в директории /boot/grub/. Если у вас несколько ОС, то содержимое меню, которое вы видите при старте системы, можно увидеть в файле /boot/grub/menu.lst. Редактировать файл можно разными способами. Если вы любите графические про-

граммы, то можно воспользоваться программой gedit, выполнив в командной строке следующую команду:

```
sudo gedit /boot/grub/menu.lst
```

В данном случае `sudo` необходимо для того, чтобы для доступа к файлу использовались права администратора, ибо доступ к файлу меню ограничен и не разрешен для простого смертного. Любители текстовых режимов могут воспользоваться утилитой vi. Посмотрим, как выглядит этот файл:

```
splashimage=(hd0,0)/boot/grub/themes/mlinux.xpm.gz
default 0
timeout 10

title Ubuntu
    root (hd0,0)
    kernel /boot/vmlinuz-2.6.27.7-9-pae
       root=/dev/disk/by-id/heredistid
       resume=/dev/disk/by-id/heredistid
       splash=silent showopts vga=0x317
    initrd /boot/initrd-2.6.27.7-9-pae

title WinXP
    rootnoverify (hd1,0)
    makeactive
    map (hd0) (hd1)
    map (hd1) (hd0)
    chainloader +1
```

В файле /boot/grub/menu.lst можно использовать много различных команд, и в данном примере мы можем наблюдать наиболее интересные из них:

- `splashimage` — задает картинку, которая будет отображаться при старте компьютера;
- `default` — система, которая будет загружаться по умолчанию по истечении определенного времени;
- `timeout` — время ожидания до начала загрузки системы по умолчанию.

После этого блоками идут описания ОС, которые установлены на компьютере и будут отображаться в меню. Наиболее интересный пункт — это параметр `title`, который задает заголовок для ОС, отображаемый в меню. Этот заголовок вы можете изменять по своему усмотрению абсолютно как угодно.

Для ОС Linux очень важным является параметр Kernel, который задает расположение ядра. Вы думаете, что он вам не понадобится и можно использовать значение по умолчанию? На первых порах это действительно так, но если вы захотите самостоятельно откомпилировать ядро с оптимизацией именно под ваше железо и включением в него только необходимых функций, то после компиляции у вас будет два ядра в директории загрузки. Возможно, вы захотите добавить еще один пункт в меню, и при загрузке выбирать между двумя ядрышками, а может захотите просто подправить путь существующего на новый. Я бы выбрал первый вариант. Лишнее ядро не будет накладным для жесткого диска с нынешними-то возможностями жестких дисков. А вот если одно ядро по какой-то причине откажется загружаться (особенно это может произойти со свежескомпилированным), то иметь второй вариант загрузки будет очень полезно.

3.2.3. 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 его можно изменить в любом текстовом редакторе, в том числе и с помощью встроенного редактора 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, разумно 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.

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

Программа login сравнивает имя пользователя со списком имен в файле /etc/passwd, а пароль — с соответствующей записью в файле /etc/shadow. Все пароли в 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 проверка происходит так сложно? Просто все пароли в файле /etc/shadow зашифрованы необратимым алгоритмом. 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 отнимет 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 оболочку (текстовую или графическую) 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/passwd. Это не очень хорошо, потому что данный файл должен быть доступен для чтения всем пользователям, так как имена пользователей требуются очень многими безобидными программами. Например, команда `ls` (просмотр файлов текущего каталога) при выполнении обращается 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/passwd остался открытым для всех, но теперь в нем уже нет пароля.

Давайте посмотрим, как выглядит файл /etc/passwd. 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 двоеточиями. Давайте разберем каждый из них:

1. Имя пользователя — login, который вы вводите.
2. Пароль — если вывод затенен, то вместо пароля будет стоять бессмысленный символ `x`.
3. UID — уникальный идентификатор пользователя.
4. GID — уникальный идентификатор группы.
5. Информация о пользователе — здесь может быть полное имя, но чаще всего значение этого параметра идентично имени пользователя.
6. Домашняя директория — каталог, принадлежащий пользователю, с которым он начинает работать при входе в систему.
7. 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, так как вместо него мы видим символ `x` (или `!!`), а сам пароль записан в зашифрованном виде 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 становится текущим после входа 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: имя пользователя 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 стоит звездочка, это запрещает соответствующему пользователю интерактивную работу с системой. Например, для пользователей bin и daemon установлены именно звездочки, значит, под этими учетными записями нельзя входить на компьютер.

3.3.2. Забытый пароль

Что делать, если вы забыли пароль, или хакер проник в систему и изменил его? Действительно, ситуация не из приятных, но все решаемо. Если у вашей учетной записи есть доступ к файлу /etc/shadow, то можно заняться его редактированием, а если нет, то посмотрим, как получить доступ из 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 большой долей вероятности гарантировать, что сохранен верный пароль. Ведь если вы случайно наберете не тот символ и не заметите этого, то не сможете воспроизвести пароль. При двойном вводе дважды допустить одну и ту же ошибку проблематично, хотя и возможно.

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 — PAM (Pluggable Authentication Modules, подключаемые модули аутентификации).

Преимущество модульной аутентификации заключается в том, что для их использования не требуется перекомпиляция программы. 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, который может принимать одно из следующих значений:
  - `auth` — модуль будет использоваться для аутентификации и проверки привилегий пользователей;
  - `account` — модуль распределения ресурсов системы между пользователями;
  - `session` — модуль поддержки сессии и регистрации действий пользователей;
  - `password` — модуль проверки пароля;

- флаг, определяющий параметры модуля. Здесь можно использовать четыре значения:
  - `required` — модуль обязателен;
  - `optional` — необязательный;
  - `sufficient` — достаточный;
  - `requisite` — модуль обязателен, а в случае ошибки управление передается приложениию;
- полный путь к файлу модуля;
- аргументы модуля.

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

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

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

3.4. 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 функционирует 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.

Процессы могут работать в двух режимах — фоновом (background) и центральном (foreground). Центральный режим для каждого терминала имеет только один процесс. Например, запустив по команде `man ls` программу для просмотра помощи, вы не сможете выполнять другие команды, пока не выйдете из программы `man`. Если ни один центральный процесс в терминале не запущен, то им является интерпретатор команд.

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. Это 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 и вы можете запустить любую программу в фоновом режиме. Для этого достаточно после указания команды через пробел поставить знак `&`. Например, выполните сейчас следующую команду:

```
man ls &
```

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

```
[1] 2802
```

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

А что же мы увидели в ответ на выполнение команды? В квадратных скобках показан порядковый номер фонового процесса, который мы запустили. 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. Это PID (Process ID, идентификатор процесса) созданного процесса, который является уникальным для всех пользователей. Это значит, что если вы запустили процесс с номером 2802, то другой пользователь никогда не сможет запустить процесс с таким же идентификатором, по крайней мере до рестарта системы. Его PID будет другим.

Запомните идентификаторы, которые вы увидели, впоследствии они 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 команду, и перед вами откроется запущенная программа man, отображающая файл помощи по использованию команды `ls`.

Раз процесс можно сделать центральным, значит можно поступить и наоборот. Чтобы вернуть процесс в фоновый режим, нажмите 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, вызвав `man имяпрограммы`.

Сервисы могут работать только в фоне и 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 (тому, что мы видели в квадратных скобках), используйте команду:

```
kill %n
```

Параметр `n` нужно заменить номером 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, необходимо 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`. Помимо этого, вы можете пожелать увидеть имя пользователя, под которым работает процесс, для этого добавьте ключ `-u`. В итоге, исчерпывающую информацию можно получить, выполнив команду:

```
ps -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 процессы можно смело убивать;
- W — не имеет резидентных страниц;
- < — обладает высоким приоритетом;
- N — имеет низкий приоритет.

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

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

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 их 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 команду 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,  248416K 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  4300 4300  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   448 S     0,0  0,2   0:00 syslogd
  530 root      15   0   436  436   376 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   468  468   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 команды 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 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-раздел, тем быстрее он работает. Конечно, пока что этот раздел практически не используется, но если перейти в графический режим и запустить пару мощных приложений, то и swap-раздела не хватит.

Программа top будет обновлять информацию о загрузке процессора с определенным интервалом времени. Для выхода из программы нажмите <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 указанное время?

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

3.5.1. Формирование задания

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

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

Дату можно и не задавать, и тогда будет взята ближайшая возможная. Если заданное время 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 половины первого. Если после этого времени забыть удалить учетную запись, то пользователь останется в системе, а это большая дыра в безопасности.

Для начала необходимо выполнить команду `at`, указав время 12:30. 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
```

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

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

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

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 и время выполнения команды, а 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 где-то 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 утра, вне зависимости от дня недели, числа или месяца.

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

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

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

Такая команда будет выполняться каждый час ровно в 00 минут.

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

Помимо этого, у сервиса cron есть несколько дополнительных директорий, упрощающих создание расписаний. 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, когда посмотрите 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 <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. 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 и напрямую, без использования команды `crontab -e`.

3.5.3. Безопасность запланированных 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 запретить, а потом позволить только то, что действительно необходимо, 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 убедитесь, что она им действительно 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, потому что не будете обращать внимание.

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. Если что-то произойдет без вашего ведома, то это можно будет определить по журналу.

3.6. Настройка сети

После установки ОС Linux легко определяет сетевые карты. С этим 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 лишь правильно его настроить.

3.6.1. Адресация

Основным для Linux является ставший стандартом для Интернета протокол TCP/IP (Transmission Control Protocol/Internet Protocol, протокол управления передачей/протокол Интернет), 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. В основном мы будем говорить о 4-й версии, которая пока что наиболее распространена.

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 *главе 11*.

   Вся адресация в сети происходит по IP-адресу, который состоит (для самой распространенной сейчас версии 4 протокола IP) из четырех десятичных чисел (октетов), 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 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, то есть 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.

   Для примера рассмотрим маску 255.255.255.0. Чтобы понять назначение маски, необходимо каждое число перевести в двоичную систему. Маска 255.255.255.0 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 в маске стоят единицы, записан адрес сети, а там, где нули — 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 является 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 так как под него в данном случае отведено одно число, максимальное значение которого 255, то нетрудно догадаться, какое количество компьютеров может быть в вашей сети.

   Рассмотрим еще пример:

   `192.168.001.001` — IP-адрес

   `255.255.000.000` — маска подсети

   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 различные сети и может передавать пакеты между ними.

В современных дистрибутивах можно найти визуальные средства настройки сетевой карты. Например, в Mandriva 2008 в GNOME такую утилиту можно найти в центре администрирования. Для его запуска выберите **Система | Администрирование | Настройка компьютера**. В окне настройки найдите раздел **Сеть и Интернет** (рис. 3.5) и здесь выберите пункт **Настройка нового сетевого интерфейса (LAN, ISDN, ADSL…)**.

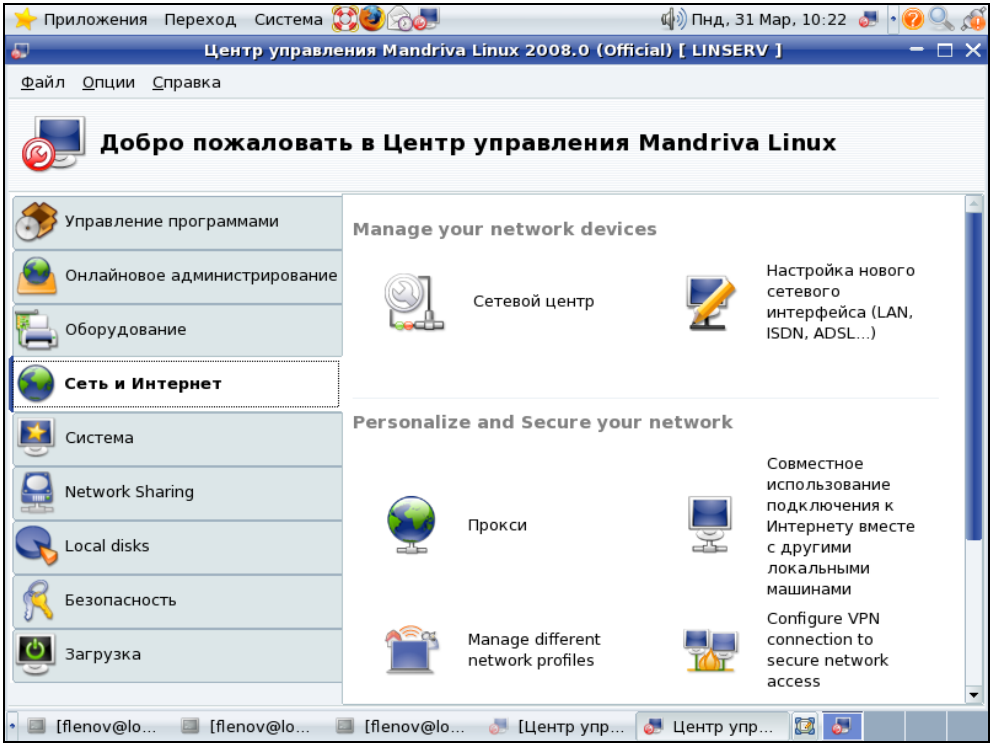


Рис. 3.5. Окно настройки сети и Интернета

Перед вами появится окно мастера установки сетевого интерфейса. На первом шаге нужно выбрать тип устройства, которое нужно настроить. Для настройки сетевой карты выберите пункт **Ethernet** и нажмите **Далее**. Дальнейшие шаги зависят от выбранного устройства и рассматривать все варианты не имеет смысла. Если вы разобрались с теорией, которую мы уже затронули, то никаких проблем с мастером не возникнет.

Мастер будет удобен и в том случае, если вы решите использовать IPv6. Этот протокол обещают внедрить уже очень долго, но конца и края процессу внедрения не видно. Поэтому я решил не тратить время и место в книге на описание данной версии. Утилиты для настройки будут использоваться те же, нужно только понимание того, как происходит адресация в IPv6, а это отдельная и достаточно длинная история. Я постараюсь написать что-нибудь на эту тему и выложить в Интернете.

3.6.2. Информация о сетевых подключениях

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

Листинг 3.1. Информация о конфигурации и состоянии сети

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

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

Как видите, в листинге 3.1 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 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 замыкается 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. Поскольку интерфейс lo создан программно (реально не существует), у него не может быть аппаратного адреса.

3.6.3. Изменение параметров сетевого подключения

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

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

Общий вид команды выглядит так:

```
ifconfig ethX параметры
```

Рассмотрим некоторые значения второго аргумента:

- `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`. Можно одновременно изменить и маску сети. Для этого выполняем

директиву `ifconfig eth0 192.168.77.3 netmask 255.255.0.0`. Здесь после ключевого слова `netmask` показана новая маска сети.

Если в момент изменения адреса сетевой интерфейс отключен, то его можно сразу же запустить командой `ifconfig eth0 192.168.77.3 netmask 255.255.0.0 up`.

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

3.6.4. Базовые настройки сети

С помощью команды `hostname` можно просмотреть имя компьютера (хоста). Выполните эту команду, и перед вами появится имя, которое вы задали во время установки. Чтобы изменить его, нужно указать новое имя в качестве параметра. Например, следующая команда устанавливает имя хоста в значение 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. Лично я к сети подключен как минимум 10 часов в день. Хотя нет, это не я подключен, а мой компьютер.

Всемирная сеть является громадным хранилищем всевозможных данных. Здесь вы сможете найти различную документацию или программное обеспечение. Впоследствии я дам ссылки в Интернете на программы, которые смогут облегчить жизнь, и их будет полезно скачать. Без доступа в сеть это невозможно.

Я не стану описывать все возможные настройки соединений, потому что конкретные нюансы вы сможете узнать у вашего провайдера.

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. Только ради этого стоит установить и настроить графическую оболочку, но пользоваться ею необходимо только на рабочей станции, но не на действующем сервере. Действующий сервер лучше держать в текстовом режиме, это намного эффективнее и безопаснее.

Для настройки Интернета в графическом режиме можно использовать ту же утилиту, что и для настройки сети. Мы ее рассмотрели в *разд. 3.6.1*.

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. 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 бьется тревога.

Хакеры, недолго думая, перешли на использование модулей к ОС Linux. Модули — это коллекции подпрограмм, выполняющие какую-то определенную системную функцию, например, поддержку устройства или, скажем,

конкретной файловой системы. Поскольку различных модулей может быть очень много, отследить хакерский модуль сложнее, а результат его использования тот же — дыра в системе безопасности. Запретив использование модулей, мы закрываем эту дыру, но при этом 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 ярда, нужно подготовиться к самому худшему, а именно — к краху системы. Да, неправильные действия действительно могут нарушить работу или сделать невозможной загрузку системы. Ядро — это основа, и если что-то указать неправильно, то ядро может работать некорректно.

Если на вашем сервере уже есть какие-либо данные, то следует сделать их резервную копию (*см. главу 13*). Помимо этого, приготовьте загрузочную дискету, которая может пригодиться 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 2.6.31. Чтобы создать загрузочную дискету для него, выполняем:

```
/sbin/mkbootdisk 2.6.31
```

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/2.6.31.

3.8.2. Обновление ядра из пакета rpm

Самый простой способ установить новое ядро — использование пакета rpm. Процесс установки такой 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 пакета rpm устанавливаются только файлы, модули и загрузчик, но чтобы можно было загрузиться с новым ядром, необходимо еще прописать ядро в загрузчик GRUB.

Установка из пакета rpm проста, но дает лишь доступ к новым возможностям 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 можем получить модульное ядро, в котором драйверы устройств могут быть как скомпилированы в одно целое с ядром, так и загружаться в виде модулей. 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 сделает невозможным обновление драйверов 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 из Интернета. Ядро всегда можно найти на сайте **www.kernel.org**.

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, то есть 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`.

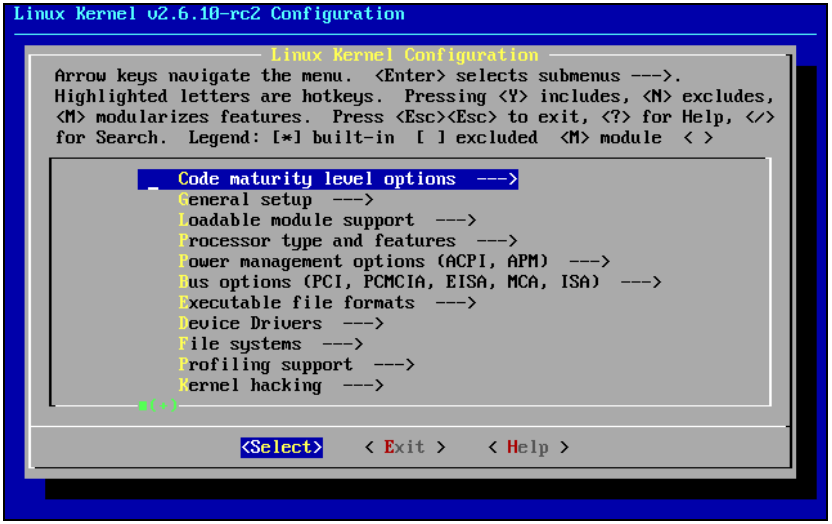


Рис. 3.6. Текстовая утилита конфигурирования ядра menuconfig

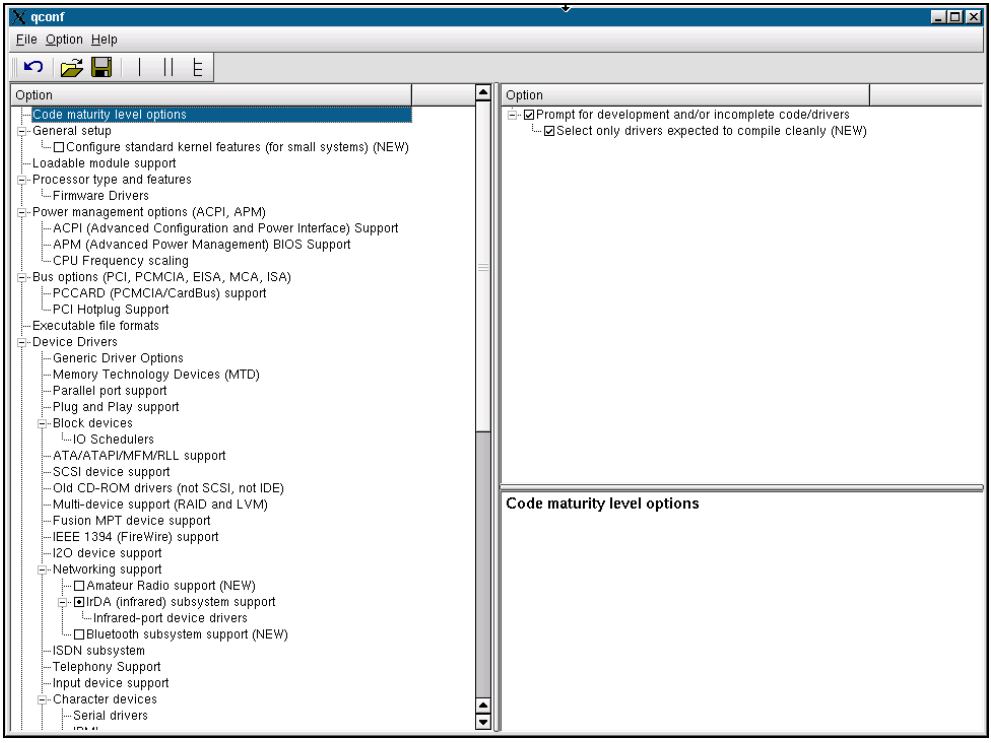


Рис. 3.7. Графическая утилита конфигурирования ядра qconf

2. `config` — сценарий, который в командном интерпретаторе задает вам вопросы о параметрах будущего ядра и в зависимости от ваших ответов формирует 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.

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 в этом параметре указывать, 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 менее 1024 Мбайт памяти, то процесс будет долгим.

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 в этом случае толку от них 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 номеру. 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

Ядро скомпилировано, и теперь его нужно добавить в загрузчик GRUB. Самый простой вариант — скопировать текущее меню и изменить его заголовок и параметр kernel, чтобы он указывал на новый файл.

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 становится известным имя и расположение файла, описание, автор, лицензия и т. д. Количество отображаемой информации сильно зависит от модуля, и если честно, в некоторых случаях она настолько скудна, что предназначение модуля остается непонятным.

modprobe

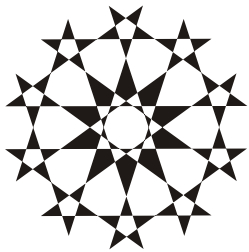
Эта команда, в основном, используется системой для загрузки установленных модулей, но можно это делать и самостоятельно. В качестве единственного параметра нужно передать команде имя модуля, который нужно загрузить.

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 действия. 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. Некоторые считают, что в Linux только один администратор root, но это не так. Root — это только имя, а администраторов может быть очень много.

Если вы являетесь администратором на предприятии, то должны следить за актуальностью записей. После увольнения сотрудника необходимо удалить его учетную запись, чтобы, недовольный увольнением, он не уничтожил важные данные. Я в своей недолгой работе администратора уже встречался с такими фактами, правда, всегда мщением занимались бывшие администраторы, но мне от этого легче не становилось.

Для работы с командами управления доступом вы должны обладать правами администратора. Для этого можно войти в систему как пользователь root или использовать команду `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:

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

После этого в каждой строке идут 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). Наличие буквы говорит о разрешении соответствующего действия.

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

Вторая строка — `drwxr-xr-x`. Это снова директория. Потом стоит тройка символов `rwx`, а значит, владельцу разрешены все операции. Следующая тройка, соответствующая группе, равна `r-x`, а стало быть, возможно чтение и исполнение, но не запись. Последняя тройка указывает, что всем остальным пользователям также доступно только чтение и исполнение.

Последняя строка в примере содержит права доступа `-rwxr-xr--` для файла (первый символ — 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 и как последовательность нулей и единиц. Если в определенном месте стоит 1 (указан один из символов r, w или x), то операция разрешена, а если 0 (указан дефис) — действие запрещено. Давайте попробуем записать права `rwxr-xr--` в виде нулей и единиц. Запишите вместо букв единицы, а вместо тире — нули. Должно получиться 111101100. Разобьем 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 было проще регламентировать доступ, предлагаю все возможные варианты значений для отдельного разряда числа:

- 0 — запрещено все;
- 1 — разрешено выполнение;
- 2 — разрешена запись;
- 3 — разрешены запись и выполнение;
- 4 — разрешено чтение;
- 5 — разрешены чтение и выполнение;
- 6 — разрешены 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--`, из которого мы получили с помощью перевода число 754. Должно выйти одно и то же. Если все получилось, значит, мы можем поднять стакан компота за здравие тех умных людей, которые придумали такую простую и удобную систему распределения прав. Да, она не идеальна, но вполне работоспособна.

> ***Внимание!***
>
> Для того чтобы иметь право создавать или удалять файлы, необходимо иметь разрешение записи на директорию. Это немного сбивает с толку начинающих администраторов, так как 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 следующих значений:

- `u` — изменить права владельца;
- `g` — изменить права группы;
- `o` — изменить права остальных пользователей;
- `a` — изменить 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` — SUID- или SGID-бит (*см. разд. 4.5*);
- `t` — sticky-бит. В этом случае только владелец файла и каталога сможет удалить его (*см. разд. 4.4*);
- `u` — всем пользователям, как и у владельца;
- `g` — всем пользователям, как и у группы;
- `o` — всем пользователям, как и у остальных пользователей.

В случае с числовым представлением команда выглядит следующим образом:

```
chmod права файл
```

Права передаются в виде восьмеричного числа из четырех разрядов:

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

   Если эти биты не надо устанавливать, этот разряд можно опустить.
2. Права пользователя. Это число может быть от 0 до 7 включительно.
3. Права группы. Это число снова может быть от 0 до 7.
4. Права остальных пользователей. Как вы думаете, какие значения может принимать параметр? Ну конечно же, от 0 до 7.

Например, мы хотим, чтобы владелец и группа имели все права (число 7) на файл text, а остальные пользователи могли его только выполнять (число 1). Команда будет выглядеть следующим образом:

```
chmod 771 text
```

Число 771 в символьном виде соответствует правам `rwxrwx--x`. Следующая команда отменит возможность чтения файла у группы:

```
chmod g-r text
```

После этой команды права доступа на файл станут `rwx-wx--x`. Теперь давайте запретим для всех 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 не даем лишних прав. Если файл не должен быть виден пользователю, то нельзя разрешать даже его чтение.

Любые лишние права могут оказаться фатальными для безопасности системы не только с точки зрения взлома, но и утечки информации. Например, файлы бухгалтерской отчетности должны быть доступны только тем, 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--`, то есть владелец может читать и изменять файл, а пользователи группы и все остальные — только просматривать. В старых системах и некоторых дистрибутивах права могут оказаться `-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`. Если выполнить ее после установки системы, будет получено значение `0022` или `002`.

Посмотрим, как маска влияет на регламентацию доступа. По умолчанию права для файлов устанавливаются в значение 666 минус маска, а для 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 – 022 = 755, а это соответствует правам `drwxr-xr-x`, то есть 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 не равно 600. Почему же так получилось? Просто вычитание происходит поразрядно, то есть 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 заменяется нулем.

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 говорили о жестких и символьных ссылках на файлы. Для начала проверим 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 жестких ссылок одинаковые дескрипторы.

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 вторая — о символьной ссылке на него. Как видите, у ссылки открыт абсолютно полный доступ. Если создать ссылку для файла /etc/shadow и не изменить ее права, то можно попрощаться с паролями, их украдут или обнулят. Помните, что любая операция по изменению файла символьной ссылки затрагивает непосредственно сам файл? Если уже забыли, то стоит запомнить или на руке выжечь.

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

4.2. Управление группами

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

И слава богу, что в Linux нельзя просто так дать пользователю право на файл, это можно только в Windows с использованием списков ACL (Access Control List, список контроля доступа). Что-то подобное пытаются реализовать в Linux, но пока подобные шутки не получили популярности.

Вместо этого можно объединить многих 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, меньшее 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):

```
#Создать группу testgroup1 с ID по умолчанию
groupadd testgroup1

#Создать группу testgroup2 с ID 506
groupadd -g 506 testgroup2

#Создать группу testgroup3 с системным ID по умолчанию
groupadd -r testgroup3
```

Вся информация о группах добавляется в файл /etc/group. Откройте его содержимое, например, в MC или наберите в командной строке:

```
cat /etc/group
```

Вы увидите содержимое файла, в самом конце которого будут 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 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 `useradd` выглядит следующим образом:

```
useradd параметры имя
```

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

- `-c комментарий` — простое текстовое описание, которое может быть любым;
- `-d каталог` — домашний каталог пользователя;
- `-e дата` — дата отключения учетной записи, после которой пользователь станет неактивным, вводится в формате ГГГГ-ММ-ДД;
- `-f число` — количество дней до отключения записи навсегда. Если указать 0, то учетная запись будет считаться некорректной сразу после устаревания пароля;
- `-g группа` — основная группа, которой будет принадлежать пользователь. Можно указывать как имя, так и идентификатор. В Red Hat каждый пользователь принадлежит какой-либо группе;
- `-G группа[,…]` — дополнительные группы, в которых будет включен пользователь. Имена групп перечисляются через запятую;
- `-m` — ключ для создания домашнего каталога пользователя. В созданную директорию будут скопированы все файлы из /etc/skel;
- `-M` — не создавать домашний каталог;
- `-r` — если указать этот параметр, то в качестве идентификатора будет выбрано число из системной области;
- `-p пароль` — зашифрованный пароль, который можно получить с помощью команды `crypt`;
- `-s программа` — командный интерпретатор, который будет обрабатывать директивы пользователя;
- `-u идентификатор` — 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 обо всех учетных записях. 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. 3.3.1*. Первый параметр — это имя. Затем стоит пароль, который спрятан в теневом файле, поэтому здесь указано `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 строке пользователя robert 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 cha-
racters from at least 3 of these 4 classes, or a 7 character long pass-
word 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 заглавных и прописных букв, цифр и дру-
гих знаков. Вы можете ввести пароль длиной в 8 символов с использованием
значений как минимум 3 из 4 указанных классов или пароль из 7 символов,
сочетающий знаки из всех классов. Пароли, содержащие часто используемые
шаблоны, будут отвергнуты.

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

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

Как видите, команда `passwd` знакомит нас с основными правилами создания сложных паролей и даже предлагает пример, который достаточно длинный

и содержит различные символы. Но я не стал бы его использовать, потому что он состоит из вполне читаемых слов. 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 все же намного меньше, чем подбор шифра вроде OLhslu_9&Z435drf. Для нахождения этого пароля словарь не поможет, а полный перебор всех возможных вариантов отнимет годы.

Но если не хочется запоминать слишком сложный пароль, то можно и объединить два слова, это будет намного лучше, чем одно простое словечко в качестве пароля.

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 ключ `-l` выводит подробную информацию. Результат выполнения такой команды должен выглядеть примерно следующим образом:

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

Обратите внимание, что в директории 5 файлов и одна директория. 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 — вроде бы не создавали. Но вспомните, в *разд. 4.2* мы говорили, что если при создании пользователя не указывается группа, то 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/robert (он здесь хозяин), но каталог выше, /home, вне его прав, он принадлежит пользователю root, так что папка .. также имеет владельца root.

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 >> /etc/skel/text
```

Здесь задается команда `ls` для просмотра содержимого текущего каталога. Потом идут 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. Но в директориях уже существующих пользователей этого файла не появится.

Эту полезную особенность я использую достаточно часто, чтобы сразу наделить нового пользователя правилами нахождения в системе, необходимыми ему файлами, документацией. Например, когда я работал на фирме в торговой компании, то в каталог пользователей автоматом помещался файл с правилами пользования компьютером, где я собрал наиболее часто задаваемые пользователями вопросы и ответы на них.

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 ее маску с помощью команды `umask`. Обратите внимание, что она равна 0002. То есть мы в *разд. 4.1* изменили свою маску, 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. Да, я бы не назвал данную проблему большим упущением в безопасности, но все же, любая предсказуемость не есть хорошо.

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
```

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, это не так. В *разд. 4.3* мы создали пользователя robert, и его идентификатор пользователя и группы был равен 501. 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.

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

- `-g` — изменить группу;
- `-b` — установить домашний каталог;
- `-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). Теперь вам не надо напрягать память или фиксировать в блокноте, 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, так как 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 нужно 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`. Этот ключ позволяет указать, членом каких групп должен быть пользователь. Можно указать несколько групп, разделенных запятыми. В данном случае только одна группа root. Для получения более подробной информации о команде `usermod` выполните команду:

```
man usermod
```

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
```

Если пользователь в этот момент находится в системе, будет выдано сообщение об ошибке.

Необходимо учитывать, что удаление пользователя не уничтожает его домашний каталог, вы должны сделать это вручную. Если же 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 удалить и ее, выполнив команду `groupdel`.

4.3.5. Настройка процедуры добавления пользователей

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

Листинг 4.1. Файл /etc/login.defs

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

```
#
#QMAIL_DIR       Maildir
MAIL_DIR       /var/spool/mail
#MAIL_FILE       .mail

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

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

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

#
# 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
```

Ряд содержащихся здесь настроек 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, но я рекомендую поставить 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` используется по умолчанию).

4.3.6. Взлом паролей

Я еще раз хочу напомнить о недопустимости использования простых паролей не только администратором, но и всеми пользователями системы. Если проследить за уязвимостями, которые находят в Linux-системах, то очень часто можно увидеть эксплоиты, которые позволяют повысить права хакера от простого пользователя до root. Если взломщик не сможет получить доступ даже в качестве простого пользователя, то и воспользоваться эксплоитом будет невозможно.

Сложные пароли должны быть абсолютно у всех пользователей. Если хакер получит доступ к файлу /etc/shadow с 1000 записями, то подбор паролей 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, перебор становится слишком продолжительным.

Подбирать сложно, если вы сравниваете полученный результат 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 1000 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 некоторой вероятностью 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 выбора 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 было нельзя, потому что внешним пользователям этого сервиса все файлы были открыты только для чтения.

Итак, если у вас хорошо настроены права доступа, то это может оказаться непреодолимой преградой для злоумышленника, даже в случае наличия уязвимости. Но нужно также иметь в виду, что не от всех уязвимостей можно защититься правами доступа, поэтому не забывайте следить за новостями и вовремя латать систему.

Мой начальник в банальном файле Excel вел на сервере план работ, в котором указывалось, что, кто и когда должен сделать. Этот файл был доступен для чтения всем, но его нельзя было изменять. Права на изменения есть только

на директорию, но не на нужный файл. Как изменить информацию в файле? Остановитесь и подумайте. Буквально через абзац я расскажу, как решить эту проблему.

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 файлов 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.

Теперь ясно, как можно изменить файл, недоступный для записи? Можно прочитать файл и сохранить его содержимое в другом файле, например, выполнить команду

```
cat имязапрещенногофайла >> имянашегофайла
```

Эта команда скопирует содержимое файла в новый. Если файл не существовал, то он будет создан из-под вашей учетной записи. Значит, вы будете его владельцем и сможете делать все, что угодно. Теперь можно удалить файл, созданный начальником, и заменить его своей копией. Так как вы — владелец, вы можете изменить этот файл, подправив нужную информацию, а потом просто изменить владельца и дату изменения, чтобы никто ничего не заподозрил.

Хитро? Ничего хитрого, но почему-то мало кто обращает на такую мелочь внимание, хотя это не мелочь, а дыра в безопасности. Бессмысленно запрещать изменение файла, если у меня полные права на директорию. Попробуйте выполнить следующие команды:

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

В первой строке переключаемся под пользователя root, чтобы от его имени выполнять команды. Вторая команда отображает файлы из текущей директории, а результат сохраняет в файл /home/flenov/1.txt. Директория выбрана не случайно, это домашний каталог пользователя flenov, в котором он может делать все, что угодно.

Следующая строка выполняет команду `ls -al` к только что созданному файлу, чтобы просмотреть подробную информацию о нем. Убедитесь, что владелец root и права доступа равны `-rwxr--r--`. Если это не так, то запретите

изменения файла для группы и для всех. В принципе, достаточно запретить изменение для всех, потому что flenov в группу root не входит, а значит, не сможет изменять файл.

Теперь переключаемся на пользователя flenov и удаляем файл. Прикол в том, что файл удалится. Единственное, что вы увидите — предупреждение, что удаляется файл, защищенный от изменений. 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`. Это 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, получив доступ к открытому диску 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 *главе 3* я уже намекал о существовании 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.

Бит SUID можно установить командой `chmod` с параметром `u+s`:

```
chmod u+s progname
```

Если теперь просмотреть права доступа к файлу progname, то они окажутся равными `-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`. Во второй группе 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, то есть 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 ее 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.

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

- `A` — не создавать метку atime записи времени последнего обращения к файлу. С точки зрения безопасности этот атрибут несет отрицательный эффект, потому что по дате обращения можно контролировать, когда файл был модифицирован. Поэтому не рекомендую устанавливать этот флаг. Но если у вас под ОС Linux работает личный компьютер, и нет необходимости отслеживать историю изменений, то можно установить этот атрибут и тем самым уменьшить количество обращений к диску (отменить лишнюю операцию записи при сохранении файла);
- `a` — позволяет открывать файл только в режиме добавления. Это значит, что уже существующие данные изменить или удалить нельзя;
- `d` — заставляет игнорировать файл при копировании. Этот флаг позволяет уменьшить размер резервной копии, но устанавливать его нужно только на файлы, не имеющие ценности и важности, например, временные;
- `i` — запрещает выполнение с файлом каких-либо действий по корректировке (изменение, удаление, переименование, создание ссылок);
- `s` — делает невозможным восстановление файла после удаления. При удалении 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 подтвердить 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 выполнил команду `lsattr`, чтобы убедиться в правильности выполнения команды. Вот теперь файл легко удалить с помощью команды `rm`.

4.7. Защита служб

В этой 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 первого издания этой книги на мой сайт было произведено несколько удачных атак, потому что из-за 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, который абсолютно бесплатен, поэтому его выбирают многие владельцы сайтов. Многие хакеры 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 форум, но это не помогло. Разработчики не устранили в последней версии одну критическую ошибку, а лишь дали инструкции по исправлению 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 дал мне ссылку на описание ошибки и способов ее устранения.

Давайте на примере абстрактного сайта **www.sitename.com** посмотрим, как происходило вторжение. На форуме можно 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 следующий 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 скрипта 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, сделав при этом все остальные разделы сервера 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 прошли через все круги ада. Рано или поздно взломщик найдет их форумы, 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, чем хакер ее найдет, то обезопасите свою жизнь.

Во время поисков уязвимых форумов я просматривал возможность получения доступа к каталогу /etc, чтобы увидеть не только количество неопытных владельцев сайтов, но и некомпетентных администраторов. Вы не поверите, но доступ открыт, наверное, на 90% серверов. Это безграмотность администраторов или их лень? Не знаю, так что точно сказать не могу. Только крупные серверы были защищены, а мелкие хостинговые компании явно экономят на заработной плате, не нанимая хороших администраторов.

4.7.1. Принцип работы

Итак, давайте посмотрим, как можно увеличить безопасность служб. Для этого создается директория, которая является для программы корневой. В Linux для этого существует команда `chroot`, которая создает окружение chroot.

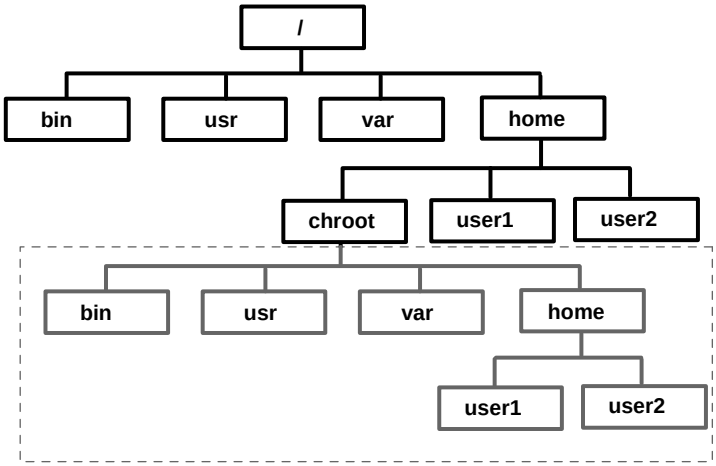


Рис. 4.1. Схема работы окружения chroot

В этом окружении создается директория, которая играет роль корневой. Выше этой директории программа, работающая в окружении chroot, попасть

не может. Получается как бы корневая файловая система внутри существующей корневой системы. 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 будет являться корнем для службы. В ней будут свои каталоги /bin, /usr и т. д., и сервис будет работать с ними, а все, что выше /home/chroot, будет недоступно.

На рис. 4.1 в рамку обведены папки, которые будут 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, что угодно.

4.7.2. Установка jail

Встроенная в Linux-систему программа chroot для создания 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 его разархивировать, 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 код открыт, и программа поставляется в таком виде.

Теперь нужно перейти в каталог jail/src (`cd jail/src`) и отредактировать файл Makefile (например, встроенным редактором MC). В самом начале файла

идет множество комментариев, и их мы опустим. После этого вы сможете увидеть следующие параметры:

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

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

Вначале задается тип ОС, по умолчанию установлен `LINUX`, а следующие три строки для FreeBSD, Irix и Solaris закомментированы. Оставим это 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, 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, в частности, 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 работы 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 перенесены, 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 указано `/home/chroot`. Параметр `home` — это домашний каталог пользователя относительно виртуальной директории. Аргумент `sh` — это командный интерпретатор, и `name` — имя пользователя, которое мы хотим добавить (оно должно уже существовать в основном окружении ОС).

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

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

У меня команда не уместилась в одну строку, поэтому я сделал перенос с помощью символа \ (который 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 уже есть. Давайте посмотрим его параметры и создадим такого же:

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

Как теперь попасть в новое окружение? Выполните команду:

```
chroot /home/chroot
```

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

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

```
ls -al /etc
```

Вы увидите всего несколько файлов, которые составляют малую часть того, что доступно в реальном каталоге /etc. Можете просмотреть файл /etc/passwd, и в нем будут только пользователи виртуального окружения. Если хакер взломает его, то он получит только эти данные и сможет уничтожить лишь

содержимое каталога /home/chroot. Вся остальная файловая система останется целой и невредимой.

Для запуска сервиса httpd нужно выполнить в виртуальном окружении команду `/usr/sbin/httpd`.

4.8. Получение прав root

Теперь у нас есть достаточно информации о разграничении доступа, 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 максимальным правам.

Так как имя пользователя ничего не значит, я рекомендую удалить пользователя root в файлах /etc/passwd и /etc/shadow, а создать новую запись с другим именем, но идентификатором, равным 0. Или просто переименовать root

во что-то более интересное. Злоумышленники, которые попытаются взломать ваш сервер, будут пытаться подобрать пароль для администратора root, но так как 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. Получив доступ под этим именем, 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, большинство хакеров — вполне здоровые люди, просто в момент взлома испытывают перенапряжение, и когда что-либо идет не так, как планировалось, хакер может запаниковать.

Именно высокое напряжение и адреналин заставляют многих молодых ребят заниматься взломом. Лично я предпочитаю получать адреналин другими методами, без нарушения закона.

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

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, где 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 уровень доступа 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 что если теперь нужно, чтобы экономист просмотрел документы бухгалтерии? Причем не все пользователи группы экономистов, а только один, и не все файлы бухгалтерии, а выборочно! Правильно решить эту задачу достаточно сложно. Если пойти простым путем и поставить на файлы бухгалтерии права `-rwxrwxrwx`, то любой пользователь сможет просмотреть бухгалтерскую отчетность, а это уже очень 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. Подход к распределению прав в Linux прост, но надежен. С другой стороны, его гибкость оставляет желать лучшего и уже долгие годы идет поиск хорошего универсального решения.

Проблему достаточно просто решить, если ввести списки ACL (Access Control List, списки контроля доступа), как это реализовано в ОС Windows. Сложность только с внедрением, так как ОС Linux не имеет соответствующего стандарта. В принципе, это ядро, на которое любой разработчик может повесить все, что угодно, и каждый производитель дистрибутива 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. К тому же, его не обновляли уже очень давно, и несмотря на первоначальные большие планы, мне кажется, что им уже не суждено сбыться.

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 забывайте, 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 редутом защиты от вторжения извне. Хакеру необходимо сначала получить доступ к компьютеру, и только если это удалось, он попытается двигаться дальше и будет пробираться до уровня файлов. Там уже действует вторая линия 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 ни получать, ни 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 нужно было срочно отправить письмо. Для этого я выполнил следующее:

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

Когда начальник вернулся, он подумал, что компьютер просто завис, и ничего не заподозрил, а я без проблем смог воспользоваться сервисом, который был мне запрещен.

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 захлебнутся уже на входе в компьютер, потому что сетевой экран не позволит сервисам даже увидеть потенциально опасные пакеты.

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

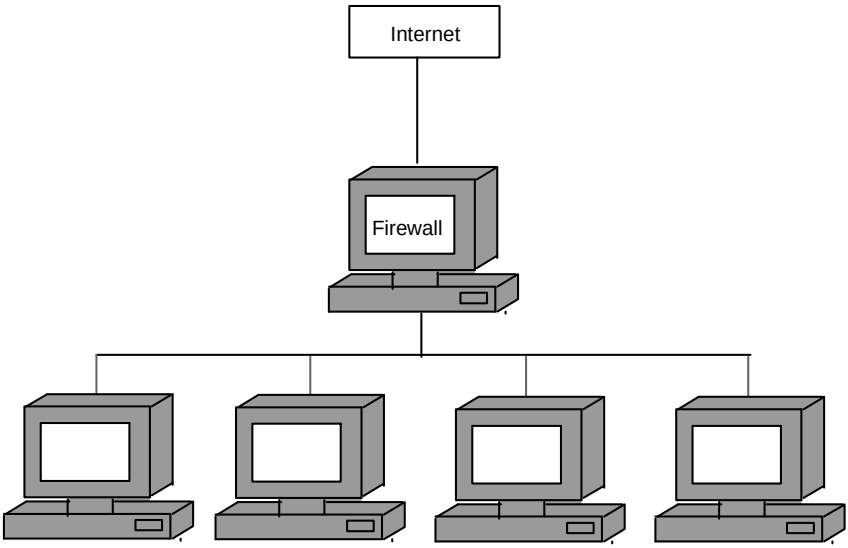


Рис. 4.2. Firewall для защиты сети

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 *главе 9*.

Так как ОС Linux уже идет с сетевым экраном в поставке, то нет смысла его отключать только потому, что есть корпоративный на границе между Интернетом и вашим компьютером. Лишнего сетевого экрана не бывает.

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. Второй немного устарел и является уже историей, но первый жив и прекрасен.

Сетевые экраны позволяют контролировать трафик, который проходит через компьютер по протоколам TCP (Transmission Control Protocol, протокол управления передачей), UDP (User Datagram Protocol, пользовательский протокол датаграмм) 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 конфигурировании сетевого экрана, 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 DoS (Denial of Service, 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 перестать 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 к необходимым им сервисам. Именно этой политики вы должны придерживаться, когда настраиваете правила для входящих пакетов.

Если двигаться от разрешения, то по умолчанию все позволено. Администратор может забыть или просто не закрыть некий вид доступа 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 строится все остальное (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, это бессмысленно, но резон есть, и он достаточно велик. Исходящий трафик могут отправлять и враги, я их перечислю:

- троянские программы, которые могут отправлять в сеть конфиденциальную информацию, или сами соединяются с хакером или с его сервером и берут 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;

- если запретить компьютерам извне инициировать соединения с вашим, то хакер может попытаться заставить ваш компьютер инициировать соединение со своим.

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 протокол передачи данных для 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 по протоколу TCP сначала устанавливается соединение с удаленным хостом, и только потом происходит обмен информацией. Благодаря этому подделка IP-адреса любого участника соединения усложняется, а иногда и становится невозможной.

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

Протокол ICMP 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 используемой вами программе, попробуйте найти другую, 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, или

статические документы HTML. Помимо этого, все программы содержат самые последние обновления и не имеют уязвимостей. Получается, что сервер безопасен? Да, но до поры до времени. Для обновления содержимого необходим какой-то доступ для 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 использовать на сервере такую политику, при которой на порт 80 будут приниматься все подключения, а FTP-сервис (порт 21) 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 нужного 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 порт 80, если FTP — то порт 21 и т. д.

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 из локальной сети, независимо от порта. Хакеры из Интернета вообще не должны иметь доступа 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 имеет смысл разрешить доступ к серверу только с IP-адресов вашего города, а остальным запретить. Но эта задача достаточно сложна в реализации.

Разделение по сетям выглядит красиво, но не всегда эффективно. Нет, я не говорю о подделке адресов, и что хакер может выдать себя за другого. Он сам может стать другим. Что я хочу сказать? Хакер может захватить один из компьютеров доверенной сети, установить на него троянскую программу и использовать для проникновения на запрещенный сервер.

Различные компании, работающие в сфере безопасности, приводят различные данные и пугают нас тем, что очень много компьютеров в Интернете находятся под контролем хакеров. Это действительно было так, но очень давно. На данный момент эти цифры не превышают и 10%.

Фильтрация нежелательных адресов

Несколько лет назад сервис **www.regnow.com** (который выступает посредником для производителей Shareware-программ, получая деньги от клиентов и обеспечивая безопасность платежей) попытался ограничить доступ с сомнительных IP-адресов. Это вполне логично. Некоторые страны кишат хакерами, и при этом число добропорядочных пользователей программ в них стремится к нулю. К таким государствам отнесли 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 недоступным, сервис запретил доступ по целым группам IP-адресов. Впоследствии выяснилось, что обойти эту систему очень просто. Хакеру достаточно воспользоваться анонимным прокси-сервером в США или Канаде, чтобы проскочить преграду. А вот у добропорядочных пользователей возникли серьезные проблемы, и они лишились возможности использовать сервис для оплаты необходимых услуг.

Из-за этих серьезных недостатков данный фильтр был снят, и разработчики **regnow.com** больше не пытаются его использовать. Просеивать все возмож-

ные 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, то есть 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 процесс зацикливался. 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 — зарезервирован для частных сетей;
- от 172.16.0.0 до 172.31.255.255 — зарезервирован 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 — зарезервирован 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 из внешней сети через сетевой экран, стоящий между Интернетом и вашей локальной сетью. Однако если сетевой экран установлен на компьютере пользователя, то все, кроме первого должно быть разрешено, иначе пользователи вашей локальной сети не смогут обратиться к вашему компьютеру. Дело в том, что зарезервированные адреса используются и должны использоваться в локальных сетях.

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.

С ОС Linux поставляется сразу две программы — iptables и ipchains. На данный момент ipchains уже практически никто не использует, а основной является iptables.

В ядре Linux определены три основные цепочки 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, то есть 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 проверка первого правила для параметров пакета даст положительный результат, поэтому будет произведен запрет и пакет удалится, и второе правило не сработает никогда.

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

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

4.10.3. Firewall — не панацея

Установив Firewall, вы не получаете полной защиты, потому что существует множество вариантов взлома, особенно если сетевой экран настроен некорректно. Мало просто установить программу, ее нужно настроить, контролировать и вовремя обновлять.

Firewall — это всего лишь замок на двери парадного входа. Злоумышленник редко пользуется парадным входом, он будет проникать в систему через черный или полезет в окно. Например, на рис. 4.2 показана защищенная сеть, а ее главная дверь — это выход в Интернет через сетевой кабель. Однако 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, и мы уже говорили, что адрес можно подделать или можно захватить доверенный компьютер.

Однажды я работал в фирме, где выход в Интернет контролировался по IP-адресу. У меня было ограничение на получение 100 Мбайт информации в месяц, а на соседнем компьютере был 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:

1. Дожидался, когда освободится нужный компьютер, например, когда хозяин уходил на обед.
2. Вытаскивал сетевой кабель на компьютере соседа, чтобы разорвать соединение.
3. Спокойно менял свой IP-адрес на соседский, и по безграничному трафику быстро качал все, что требовалось.
4. После скачивания возвращал IP-адрес и кабель на место.

Таким образом я в течение месяца получал необходимую мне информацию.

Дальше стало еще проще. Я установил прокси-сервер на соседний компьютер и стал его использовать. После этого мы всем отделом заходили в Интернет через один IP-адрес, имеющий неограниченный трафик.

В современных сетевых экранах простая замена IP-адреса не позволит 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 защищен сетевым экраном, и в нем разрешен только порт 80. А хакеру больше и не надо!!! Ведь это не значит, что нельзя будет использовать другие протоколы. Можно создать туннель, через который данные одного протокола передаются внутри другого. Так появилась знаменитая атака Loki, которая санкционирует передачу команды для выполнения на сервере через ICMP-сообщения Echo Request (эхо-запрос) и Echo Reply (эхо-ответ), подобно команде `ping`.

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 руководствуется только заложенными в программу алгоритмами. 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 (сниффера) и предъявить подделанные параметры идентификации сетевому экрану. Так было взломано немало систем.

Управление паролями должно быть четко регламентировано. 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 охранник, который спасет от воров и мелких грабителей. Но если подъехать к такой организации на танке, то эта охрана не поможет.

Сетевой экран — это как охрана на дверях, защищает от мелких хакеров, которых подавляющее большинство. Но если банком займется профессионал, то он может добиться успеха. Я не говорю, что обязательно добьется, но вполне возможно.

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 сейфы со сложными механизмами защиты, и если вор не знает, как их обойти, он 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 расшифровать 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. В Mandriva 2008 для этого используется уже знакомая нам утилита Центр управления. В ней нужно перейти в раздел **Безопасность** и щелкнуть по иконке **Set up your personal firewall** (рис. 4.3) или **Настроить персональный сетевой экран**.

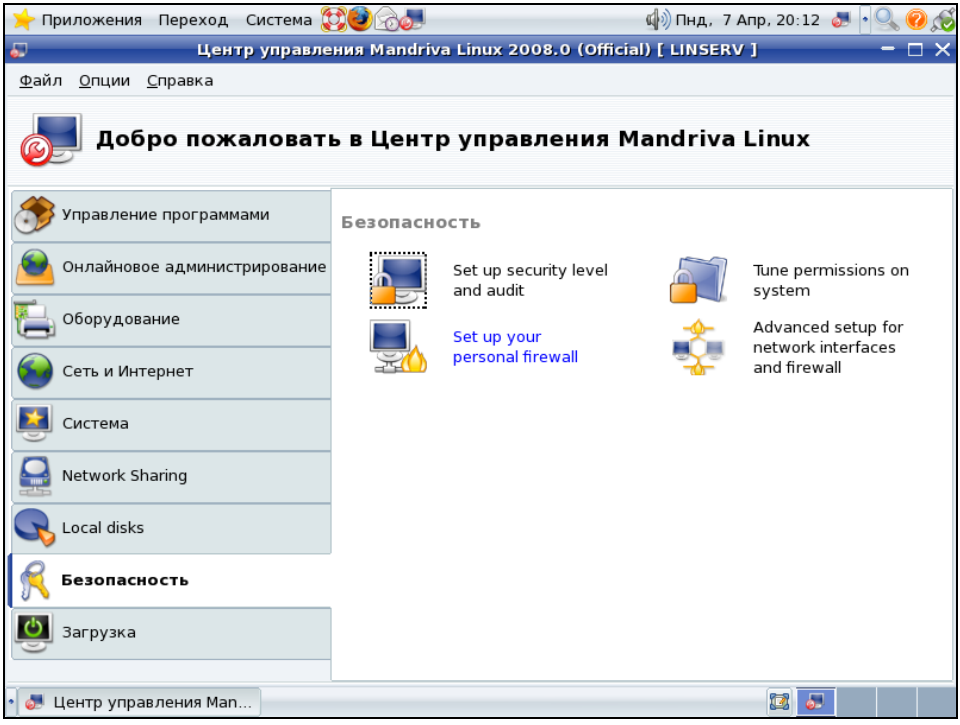


Рис. 4.3. Окно запуска графической утилиты конфигурирования сетевого экрана

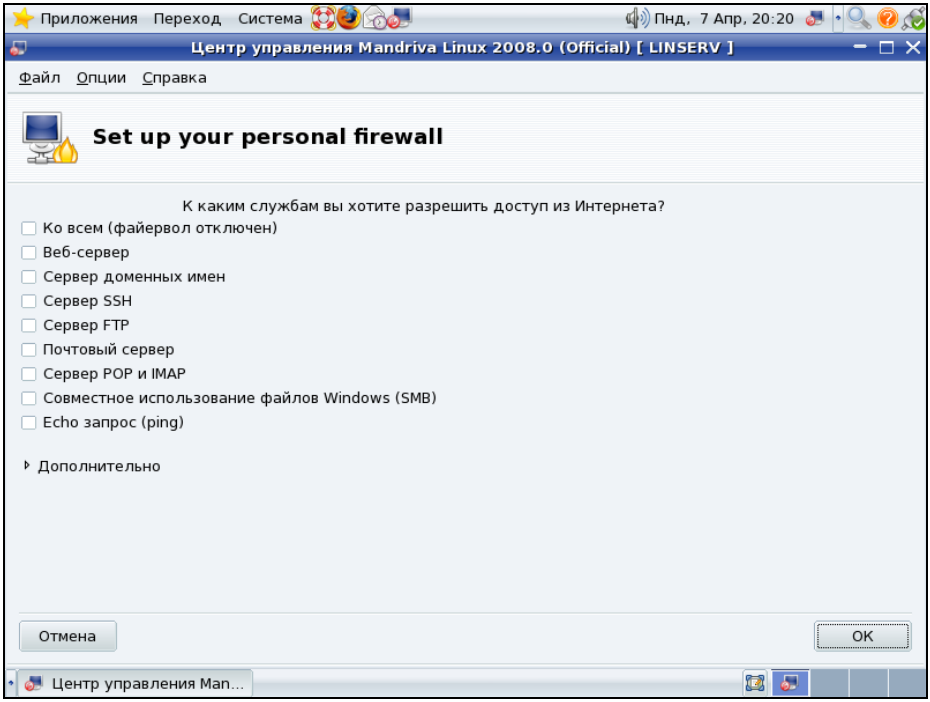


Рис. 4.4. Выбор сервисов, к которым должен быть предоставлен доступ

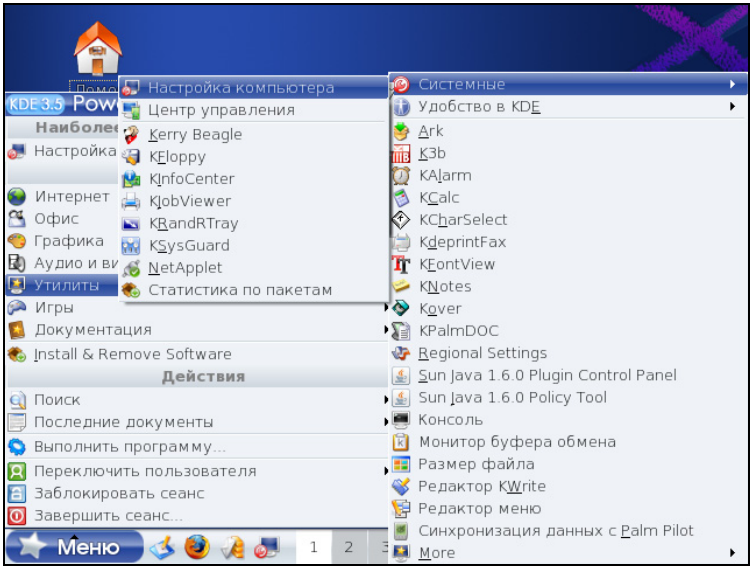


Рис. 4.5. Запуск центра управления в KDE

Визуальную настройку упростили, дальше просто некуда. Запустите утилиту конфигурирования, перед вами появится окно (рис. 4.4), в котором достаточно выбрать, к каким сервисам, установленным на компьютере, необходимо разрешить доступ по сети. Самый опасный пункт — первый. Если вы выберите его, значит, вы абсолютно ничего не поняли из того, что я пытался рассказать в этой книге ранее.

Кстати, я все это время говорю, как работать с GNOME, но до сих пор не показал, где находится утилита конфигурирования в другой распространенной графической оболочке Linux — KDE. Здесь центр управления запрятан в **Меню | Утилиты | Системные | Настройка компьютера** (рис. 4.5).

Если вы не настраивали никаких правил во время установки Linux, то советую проскочить сейчас взглядом по списку сервисов и разрешить доступ только к тому, что реально будет использоваться по сети.

4.11. *ipchains*

Данный сетевой экран был стандартом де-факто лет десять назад, но за последние четыре года он потерял в популярности, потому что появилась хорошая альтернатива, предоставляющая более надежную защиту — iptables. Именно поэтому из третьего издания данный раздел удален.

Но если вас все же интересует эта тема, вы можете почитать достаточно подробное руководство по использованию ipchains. Его вы найдете в директории Doc на прилагаемом компакт-диске. А чтобы не тратить лишнее время, давайте перейдем к следующему разделу, где будет рассматриваться современный iptables.

4.12. *iptables*

Сетевая система ядра Linux называется netfilter и предоставляет нам фильтрацию пакетов с сохранением состояния и без него, а также NAT и IP Masquerading. Для управления сетевым ядром используется утилита командной строки iptables, а для управления IP 6-й версии используется утилита ip6tables. Эта утилита схожа по синтаксису с ipchains, но предоставляет нам больше возможностей, например, журналирование.

Если сервис iptables у вас еще не запущен, то следует сделать это сейчас. Для этого выполните команду:

```
service iptables start
```

Для остановки сервиса необходимо заменить последнее слово в команде на `stop`.

Чтобы заставить сервис запускаться автоматически при старте системы, следует выполнить команду:

```
chkconfig -level 345 iptables on
```

Сама команда имеет в простейшем случае такой вид:

```
iptables команда [ключи]
```

Команда в большинстве случаев включает в себя указание цепочки, то есть набора правил, которые применяются в определенных случаях. В систему встроены несколько цепочек, которые постоянны и не могут быть удалены. Вот основные из них:

- INPUT — входные данные;
- OUTPUT — выходные данные;
- FORWARD — перенаправление.

Давайте посмотрим на пример, который разрешает любой трафик на локальном интерфейсе loopback. Этот интерфейс есть на каждом компьютере, и указывает на вашу машину. Он безопасен, ибо по сети этот интерфейс никто не видит, поэтому на нем можно разрешить все. Итак, команда выглядит следующим образом:

```
iptables -I INPUT 1 -i lo -p all -j ACCEPT
```

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

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

4.12.1. Основные возможности *iptables*

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

- `-A цепочка правило` — добавить правило в конец цепочки. В качестве параметров указывается имя цепочки (`INPUT`, `OUTPUT` или `FORWARD`) и правило;
- `-D цепочка номер` — удалить правило с указанным номером из заданной цепочки;
- `-R цепочка номер правило` — заменить правило с указанным номером в цепочке;

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

Если вы знакомы с ipchains, то должны были заметить невероятное сходство. Но есть и очень существенные отличия. Например, с помощью ключей `-o` и `-i` очень просто указывать, с какого и на какой интерфейс направляется пакет.

Сразу же необходимо заметить, что для сохранения цепочек iptables нужно выполнить команду:

```
service iptables save
```

Если не сохранить правила, то они будут действовать только до первой перезагрузки, после которой настройки будут сброшены, и все придется настраивать заново. А оно вам нужно?

Теперь пора переходить к практическим примерам конфигурирования сетевого экрана. Ключ `-P` используется для указания значения по умолчанию. А что должно быть по умолчанию для максимальной защиты? Конечно же,

запрет. Чтобы запретить любые входящие, исходящие и переадресуемые пакеты, выполняем команды:

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP
```

Рассмотрим эти команды подробнее. После имени команды идет ключ, который указывает, что мы делаем. Ключ `-P` говорит, что мы задаем значение по умолчанию. После этого должно идти имя цепочки, куда добавляется правило. В первом случае это цепочка `INPUT`, а значит, правило будет действовать на входящие данные. Во втором — `OUTPUT`, то есть исходящие данные, а в третьем случае — `FORWARD`. Последний параметр говорит о том, что нужно делать с пакетами, удовлетворяющими условиям, которые мы перечислили. Слово `DROP` говорит о необходимости удаления.

Давайте сделаем небольшое лирическое отступление, дабы понять разницу между `REJECT` и `DROP`. В качестве последнего параметра (действия) можно указать вместо `DROP` значение `REJECT`, что тоже будет являться запретом. Разница в том, что в случае `REJECT` отправитель запроса получит уведомление о том, что запрошенный доступ для него запрещен. Это не есть хорошо: ведь это лишняя информация, которую хакеру не нужно знать. При ключе `DROP` полученный запрещенный пакет будет просто удален без каких-либо уведомлений и предупреждений.

Если стоит выбор между `REJECT` и `DROP`, а вы не знаете что выбрать, я рекомендую выбирать второе.

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.

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

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

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

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

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

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

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

Следующий пример разрешает подключения по SSH:

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

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

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

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

Следующая команда запрещает доступ к FTP-порту:

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

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

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

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 порту 80, но контролировать, чтобы пакеты содержали только допустимые параметры. К безопасности WEB-сервера мы вернемся в *главе 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 также учесть аспект защиты информации. Допустим, что у вас есть сервер, который принимает закодированный трафик с помощью stunnel (Security Tunnel, безопасный туннель: создание шифрованного канала между двумя машинами, которое мы 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.

4.12.2. 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, то есть включаем маскарадинг.

Кроме того, можно использовать NAT (Network Address Translation, трансляция сетевых адресов). Для чего нужна трансляция? Сетевых IP-адресов на всех не хватает, поэтому локальные сети используют адреса из зарезервированного диапазона (10.x.x.x или 192.168.x.x). Эти адреса не маршрутизируются и не могут использоваться в Интернете. Но как же тогда использовать

в таких сетях Интернет? Все пакеты, направляемые на адреса в Интернет, перенаправляются на шлюз, на котором работает NAT. С помощью NAT адрес компьютера из локальной сети подменяется интернет-адресом, и пакет продолжает свой путь из локальной сети во всемирную. Таким образом, достаточно иметь только один интернет-адрес, который присваивается шлюзу, и вся локальная сеть сможет работать с ресурсами всемирной сети.

Для настройки NAT команда может выглядеть следующим образом:

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

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

```
modprobe iptable_nat
```

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

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 жесткие запреты сделают вашу систему надежной, иначе она будет подвержена рискам.

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 этом 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. Если вам уже не нужен больше доступ по FTP, вы захотите его убрать. Отключив FTP-сервер, не забудьте удалить разрешающие правила из 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, и сервер попал под угрозу. Хорошо, что IP-адрес 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 (например, основной шлюз), все же имели жестко закрепленный адрес. Это предотвратит и случайное попадание IP-адреса в недобросовестные руки, и потерю привилегий теми, кто в них нуждается.

Чтобы четко контролировать IP-адреса, желательно использовать DHCP-сервер, но жестко фиксировать адреса тем, кто нуждается в привилегированном доступе и имеет фильтры в цепочках.

Очень распространенная уязвимость — забывчивость администратора. Настроив соединение, администратор банально забывает сохранить таблицу цепочек, и после перезагрузки сервера все настройки теряются.

При формировании правил будьте очень внимательны. Некоторые сервисы (такие как FTP), могут требовать для своей работы более одного порта. Если вы не откроете все нужные порты, то можете не получить необходимого результата.

При настройке сетевого экрана из графической оболочки будьте особенно осторожны. При неправильных запретах X Window 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. Если сервер корпоративный и рабочий, то телефон расплавится от звонков возмущенных пользователей.

Тестируйте все используемые соединения после каждого изменения конфигурации сетевого экрана. После внесения нескольких модификаций найти ошибку очень тяжело.

Для поиска конфликтных цепочек я сохраняю конфигурацию во временный файл 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. Я и сам регулярно путаюсь.

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 (forward) и, наконец, выход, то есть цепочку output. Таким образом можно 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 порт 80 пакетов, то в результате этого порт 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. Это касается сетевого экрана, который устанавливается на границе вашей сети и Интернета, а не персонального экрана, который стоит на каждом компьютере.

Регулярно делайте резервную копию цепочек. Для этого можно сохранять их содержимое в отдельном месте (желательно, на другой компьютер или на сменный носитель). Тогда в случае возникновения проблем можно быстро восстановить работающие 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 полную безопасность, и нет такого правила, которое нельзя обойти.

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

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

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

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

Самое неприятное происходит, если сетевой экран настроен на отправку сообщений об ошибках. 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. Некоторые сетевые экраны анализируют только первые блоки в сессии, а все остальные считают правильными. Логика такого поведения 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.

Данной уязвимости были подвержены даже некоторые экраны, используемые в Linux, но ошибка уже давно исправлена и сейчас так обойти Firewall нереально.

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. Этот сервис работает на двух портах: ftp и ftp-data. Пользователь подключается к серверному порту ftp для выполнения команд, но когда вы запрашиваете получение файла, сервер сам инициирует соединение с клиентом, а этого сетевой экран не разрешит. Для FTP эту проблему решили, добавив возможность работы в пассивном режиме, но в других программах (например, в чатах) вопрос остается открытым. Тут его решают по-разному, в зависимости от производителя программы.

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 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 серьезной 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-сниффера 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. Мы не можем гарантировать спокойную жизнь, пока не можем четко знать источника и конкретного человека. С другой стороны, изменение ситуации может повлиять на нашу свободу.

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 администраторы 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 пропускает ICMP-пакеты, то сканирование можно провести с помощью `traceroute`. Возможно, появится сообщение об ошибке. В данном случае строка 18 сообщает о превышении времени ожидания ответа. Это значит, что пакет отправлен, но сервер отбросил запрос, а значит, вероятнее всего пакет с TTL, равным 18, был уничтожен без предупреждения и информирования отправителя.

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-й пропадет, а 19-й пройдет дальше в сеть, потому что на сетевом экране такой пакет появится с TTL, равным 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. Может, он просто не запрещает ICMP-трафик.

Внутреннюю сеть можно просканировать и через DNS-сервер, если он находится внутри нее и доступен для всеобщего использования. Но это уже отдельная история, не касающаяся сетевого экрана.

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.

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, с которого доступ разрешен.

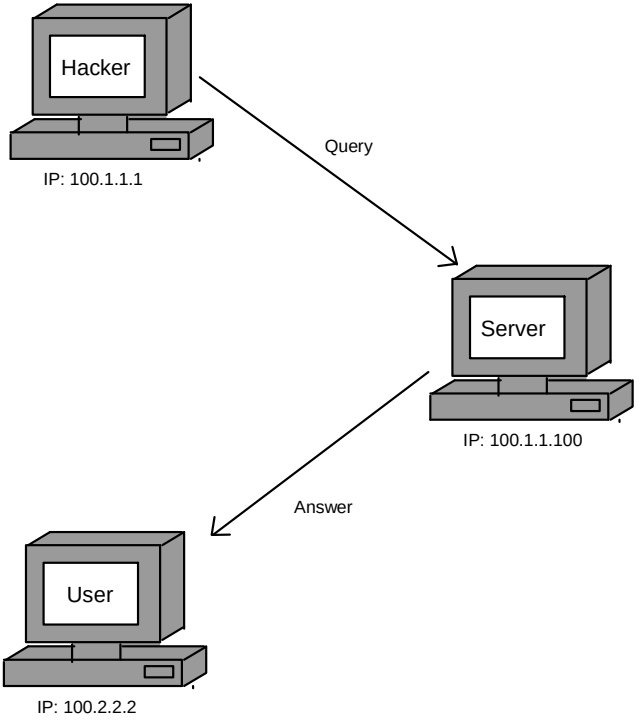


Рис. 4.6. Подмена IP-адреса

Но это еще не все. Просто воспользовавшись чужим именем, хакер ничего не добьется: он не увидит отклика сервера (рис. 4.6). 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.

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 может показаться сложной, потому что требует знания необходимых портов, но этот способ наиболее надежный, и для построения реальной защиты рекомендуется использовать именно его. А вот для решения простых задач есть другой метод — использование файлов

/etc/hosts.allow и /etc/hosts.deny. Первый файл содержит записи хостов, которым разрешен доступ в систему, а во втором прописаны запреты.

При подключении к серверу файлы проверяются следующим образом:

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

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 в файле hosts.allow 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.
```

Эта строка разрешает 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 любой сетевой экран позволяет осуществлять куда более тонкие настройки.

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

Почему нежелательно играть с цепочками сетевого экрана? Случайное удаление или добавление ошибочной записи может нарушить работу сервера или понизить его безопасность. Именно поэтому 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 я 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
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100024    1   udp  32768  status
100024    1   tcp  32768  status
391002    2   tcp  32769  sgi_fam
```

Как видите, одна команда может сообщить весьма большое количество информации, поэтому 111 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*, 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 воспользоваться ею 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.

Если у вас в системе нет пользователя, не обладающего правами администратора, то добавьте его сейчас (*см. разд. 4.3*). 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, 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 участвует 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 пользователь robert сможет от имени root просматривать любые файлы в системе, включая 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. Тут желательно указывать 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, так как придется отдавать пароль администратора пользователю с учетной записью 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.

Желательно не передавать 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 команды, а указываете ключевое слово `ALL`:

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

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

С помощью утилиты `sudo` можно предоставить доступ для корректировки файлов. 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, то есть администратора системы. Версии sudo от 1.5.7 до 1.6.5.p2 содержат ошибку выделения памяти. Хакер может воспользоваться этим для выполнения атаки переполнения стека. Проверьте вашу версию, вызвав команду `sudo` с ключом `-V`. Если вы являетесь администратором, то увидите на экране подробную информацию о программе, вроде приведенной 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 приводить результат выполнения команды полностью, 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 указывается 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.

4.17. Привилегии пользователя

В *разд. 4.9* мы говорили о том, что отсутствие стандартной гибкой возможности распределения прав является узким местом. Возможности есть, и в Интернете можно найти много решений. Кое-что интересное уже появилось в графической утилите управления пользователями в Ubuntu. На рис. 4.7 показана закладка привилегий в утилите настройки пользователей.

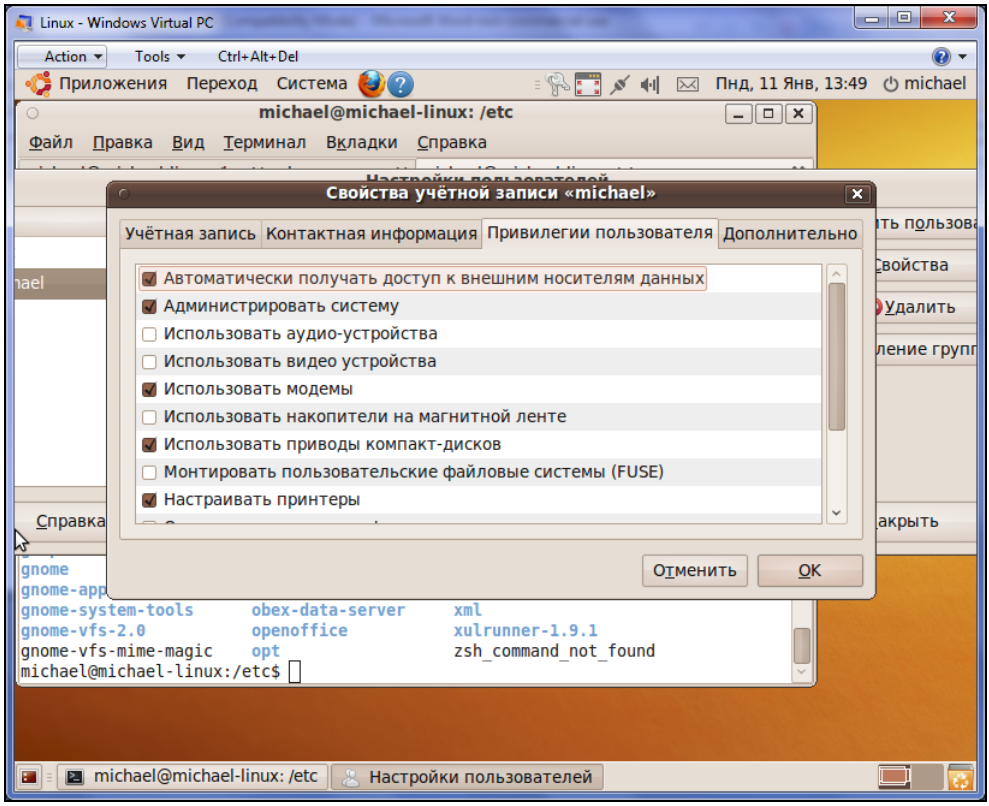
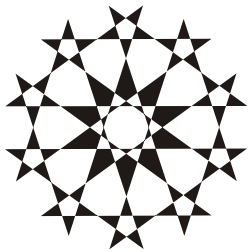


Рис. 4.7. Закладка привилегий пользователя в Ubuntu

Это далеко не самые мощные возможности, но уже что-то. Если пользователь не должен иметь доступа к функциям администрирования системы, то их нужно отключить. Без этих функций пользователь и запущенные им программы смогут нанести системе меньше вреда, если вообще смогут сделать что-то злостное.

Далеко не всегда пользователи намеренно делают что-то плохое на серверах. Очень часто это происходит случайно, по незнанию или неопытности. Каждый лишний уровень защиты между пользователем и системой — это наш спокойный сон.



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.

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.

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

Для начала 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. Сетевые соединения

Графическая утилита настройки сети в Ubuntu далека от идеала и наглядности. Подобные утилиты в Mandriva или в Suse намного интуитивно понятнее в работе. Но Ubuntu почему-то самый популярный. Для того чтобы начать настройку соединений, выберите **Система | Параметры | Сетевые соединения**. В результате появится окно, в котором по разделам разбиты все сетевые устройства, которые могут вывести вас во внешний мир. Выберите устройство,

параметры которого хотите настроить, и нажмите кнопку **Изменить**. Перед вами должно открыться окно, показанное на рис. 5.1 (если производители ничего не изменили).

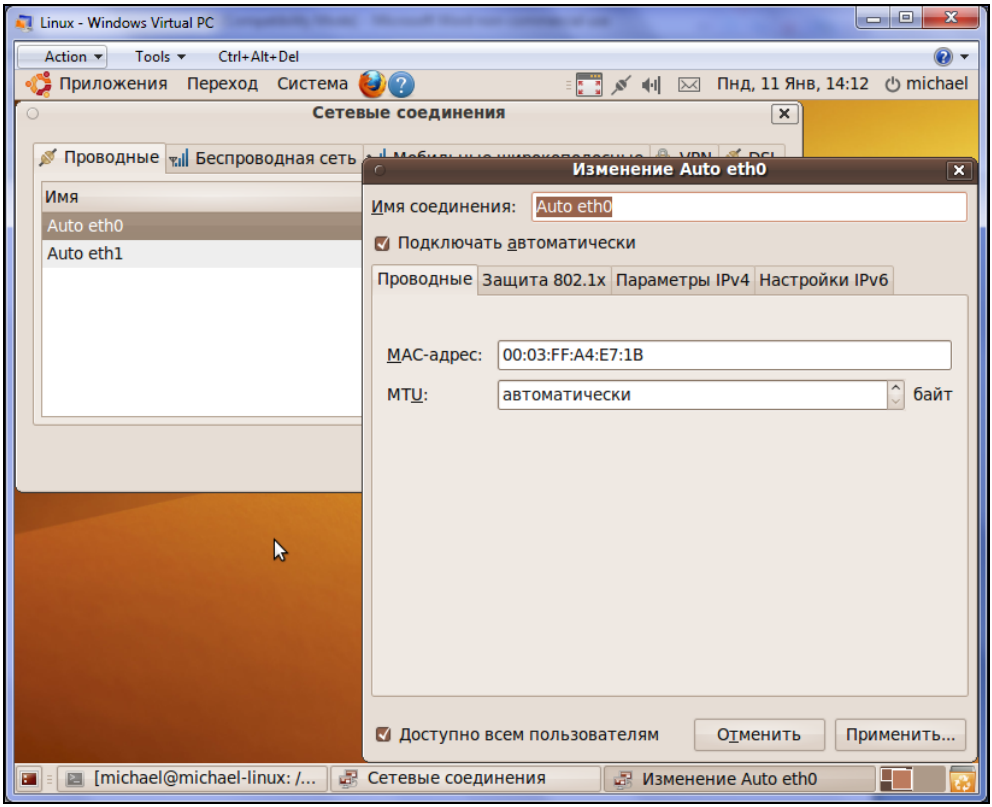


Рис. 5.1. Графическая программа настройки сети

В этом окне и проводятся настройки сетевого соединения, параметры которого должны быть понятны, если вы знакомы с принципами работы сетевых протоколов, а если что-то непонятно, то поищите в алфавитном указателе в конце книги или прочитайте до конца эту главу.

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. О таких запросах мы уже говорили, когда рассматривали сетевой экран (*см. разд. 4.10*).

Например, выполните команду `ping 195.18.1.41`, и в ответ вы получите 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
```

В данном случае IP-адрес нужно заменить любым адресом в вашей сети или в Интернете, например:

```
ping microsoft.com
```

Первая строка информационная, в ней отображается IP-адрес компьютера, с которым будет происходить обмен сообщениями (если вы указали символьное имя хоста, то простой командой `ping` можно узнать его IP-адрес). В конце строки указан размер пакетов данных, которые будут отправляться.

Следом на экране начнут появляться строки 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 происходить из-за неустойчивой работы оборудования, плохого кабельного соединения или в случае, когда один из маршрутизаторов в сети между компьютерами выбрал неправильный путь, и пакет не достиг места назначения;
- `ttl` — время жизни пакета. При отправке пакета ему отводится определенное время жизни, которое задается целым числом. По умолчанию в большинстве версий `ttl` равен 64, но 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 ее стабильности, если значение параметра не сильно изменяется от пакета к пакету. Но учитывайте, что время, затра-

ченное на отправку и получение первого пакета, почти всегда выше, особенно при указании имени, а не IP. Это связано с затратами на поиск в базе DNS. Все остальные пакеты должны идти равномерно.

Если ответ на какой-нибудь запрос не был получен, то вы увидите сообщение о его потере. Дождитесь, пока программа не отправит 7—10 пакетов, чтобы по их параметрам оценить качество связи, после чего можно прерывать сеанс нажатием клавиш <Ctrl>+<C>. В результате вы увидите краткую статистику обмена сообщениями: количество отправленных, полученных и потерянных пакетов, а также минимальное, среднее и максимальное время обмена пакетами.

Вот основные параметры, которые можно использовать в команде `ping`:

- `-c n` — завершение работы после отправки (и приема) `n` пакетов. Например, вы хотите послать пять запросов, тогда команда будет выглядеть следующим образом: `ping -c 5 195.10.14.18`;
- `-f` — форсированная передача. Например, вам нужно быстро отправить 50 запросов, тогда команда будет выглядеть 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` — размер пакета. Например, если вы хотите отправить пакет размером в 1000 байт, команда должна выглядеть так: `ping -s 1000 195.10.14.18`. В некоторых старых версиях ОС были ошибки, и при получении слишком большого пакета система зависала. В настоящее время погрешности такого рода отсутствуют, и встретить подобную систему в Интернете сложно.

Это наиболее часто используемые параметры. Дополнительную информацию по команде можно получить в справочной системе, выполнив команду `man ping`.

> ***Внимание!***
>
> Не каждый сервер отзывается 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 компьютеру. Результат ее выполнения имеет примерно следующий вид:

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address    Foreign Address      State
tcp        0      0 FlenovM:ftp      192.168.77.10:3962   ESTABLISHED
tcp        0      0 FlenovM:ftp-data 192.168.77.10:3964   TIME_WAIT
```

Информация представлена в виде таблицы. Давайте рассмотрим ее колонки:

- `Proto` — базовый протокол, который использовался для соединения. Чаще всего здесь можно видеть `unix` или `tcp`;
- `Recv-Q` — количество байтов, не скопированных пользовательской программой из очереди;
- `Send-Q` — количество байтов в очереди на отправку удаленному компьютеру;
- `Local address` — локальный адрес формата `компьютер:порт`. В качестве порта может использоваться как символьное имя, так и число. В приведенном 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 множество дополнительных параметров, и полное их описание можно увидеть в файле справки, набрав команду `man netstat`.

В случае непредвиденной ситуации и подозрения на проникновение в систему извне вы с помощью этой команды сможете определить, через какой сервис произошло вторжение, и что хакер в 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. *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, за тысячи километров. Нужно лишь подключиться к компьютеру на порт Telnet-сервера с помощью Telnet-клиента, и можно считать, что вы в системе и используете мощности производительной техники, которая может быть недоступна домашнему пользователю.

В Windows очень мало утилит, способных работать в командной строке, поэтому в этой ОС необходим (и широко используется) графический режим. Да и сама командная строка в Windows 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 медленных каналах связи.

Командная строка Linux по сравнению с графическим режимом практически не расходует трафик и может приемлемо работать даже по самым медленным

каналам, например, по соединениям GPRS сотового телефона или домашнего модема, где скорость достаточно мала.

Как вы уже поняли, программное обеспечение Telnet состоит из серверной и клиентской частей. При запуске Telnet-сервера открывается порт 23, к которому можно подключиться с клиентского компьютера и выполнять любые команды, которые позволяет удаленный сервер.

Но это не все, с помощью Telnet-клиента можно подсоединяться и к любому сервису, протокол которого является текстовым (то есть команды читаемы и представляют собой текст). Например, можно подключиться на порт 25 и прямо из командной строки отправить E-mail-сообщение, подавая команды SMTP-сервера.

Если у вас есть установленный FTP-сервер, то уже сейчас вы можете выполнить команду:

```
telnet localhost 21
```

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 реальным компьютером. Поэтому на всех боевых серверах отключайте сервер Telnet. Он небезопасен, потому что данные передаются в открытом виде и могут быть перехвачены. Все попытки сделать Telnet защищенным заканчивались неудачно и не приводили к желаемому результату. Один из относительно безопасных вариантов использования Telnet — это подключение через канал шифрования OpenSSL (от Secure Socket Layer, протокол защищенных сокетов). Но наибольшую популярность получило управление сервером через протокол SSH (OpenSSH, Open Secure Shell), который мы рассмотрим позже, в *разд. 5.3*.

Таким образом, Telnet-клиент в системе необходим, а Telnet-сервер, если он у вас установлен, следует срочно удалить и забыть как страшный сон.

Если вы все же решили использовать Telnet-сервер, то это необходимо делать только через протокол для безопасной связи по сети с применением открытых и секретных ключей (*см. разд. 5.2*). 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`. Если сервер запущен и под-

ключение разрешено сетевым экраном, то перед вами появятся похожие 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 — 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 дистрибутива Red Hat 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 в приветствии снова 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
```

Но главная проблема — не болтливость, а отсутствие безопасности. Утилита передает все команды в открытом виде, без шифрования. Если хакеру удастся каким-либо образом перехватить ваши пакеты, то он сможет увидеть не только выполняемые вами команды, но и пароли доступа.

5.1.5. r-команды

В Linux есть так называемые r-команды: `rlogin`, `rsh`, `rcp`, `rsync`, `rdist`. Мы не будем их рассматривать, потому что все они создают большие проблемы в безопасности. Если Telnet-клиент полезен для тестирования сервисов, то эти команды я включил в обзор только для того, чтобы вы удалили их из системы, исключив тем самым соблазн их использовать и возможность их применения хакером.

Все r-команды устарели и небезопасны, потому что позволяют подключаться к системе удаленно и при этом передают данные без шифрования.

5.2. Шифрование

Во времена рождения Интернета и первых сетевых протоколов еще не задумывались о безопасности. О ней стали думать только тогда, когда начали происходить реальные взломы и целые эпидемии. Одним из самых больших упущений было то, что в большинстве протоколов данные по сети передаются в открытом виде, а сетевое оборудование позволяет прослушивать сетевой трафик.

В локальной сети есть свои особенности. 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: или все компьютеры объединяются напрямую в одну общую шину, или они соединяются в 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 приходит ко всем компьютерам сети, а установленные на них сетевые карты проверяют адресата: если пакет направлен им, то принять, иначе — пропустить.

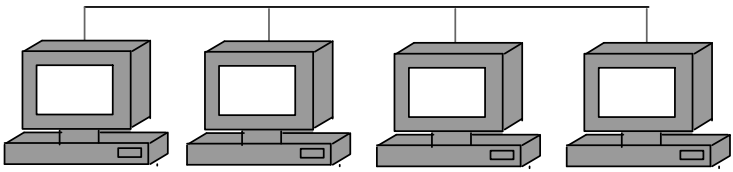


Рис. 5.2. Схема соединения компьютеров "Общая шина"

Компьютер обрабатывает только свои пакеты, но сетевая карта может видеть абсолютно все, что через нее проходит. И если захотеть, то, воспользовав-

шись утилитой для прослушивания трафика (снифферером), можно просмотреть все данные, проходящие мимо сетевой карточки, даже если они предназначены не вам. А так как 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 (switch) используется топология звезда (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 и более.

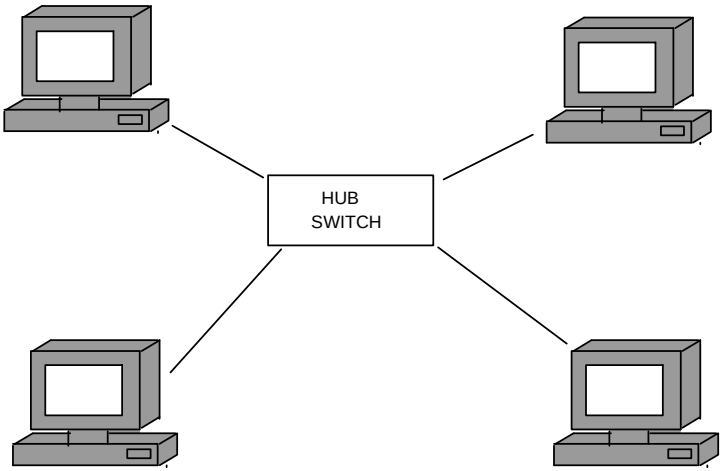


Рис. 5.3. Схема соединения компьютеров "Звезда"

Если в центре стоит хаб (простое и дешевое устройство), то все пакеты, пришедшие с одного компьютера, копируются на все узлы, подключенные к этому хабу. Таким образом, любой компьютер тоже может видеть пакеты других участников сети.

В случае с коммутатором (более интеллектуальное и более дорогое устройство) пакеты будет видеть только получатель, потому что коммутатор имеет

встроенные возможности маршрутизации, которые реализуются в основном на уровне MAC-адреса (Media Access Control Address, адрес управления доступом к среде), который называют физическим адресом. Маршрутизация — это слишком громко сказано, она доступна только в дорогих устройствах, а в более дешевых есть только коммутация. Физический адрес — это 48-разрядный серийный номер сетевого адаптера, присваиваемый производителем. Он уникален, потому что у каждого изготовителя свой диапазон адресов (однако не стоит думать, что его невозможно подделать). 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, поскольку это накладно и в некоторых случаях просто невыполнимо, потому что потребует изменения всех существующих программ. Но было найдено более удобное и универсальное решение — туннелирование (tunneling), которое позволяет программам удаленного доступа различных разработчиков взаимодействовать друг с другом, а также поддерживает несколько методов проверки подлинности, сжатия и шифрования данных. В общих чертах, туннелирование выглядит следующим образом (на примере FTP):

- на клиентском компьютере на определенном порту (Порт 1) вы должны запустить программу для шифрования трафика. Теперь, вместо того чтобы передавать данные на удаленный компьютер, вы должны с помощью FTP-клиента соединиться с Портом 1 своего компьютера и направлять данные на него. Полученные данные будут шифроваться и передаваться по сети в закрытом виде программой шифрования;
- 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.

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 в этот момент 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 в зашифрованном виде передает на сервер данные и получает ответы.

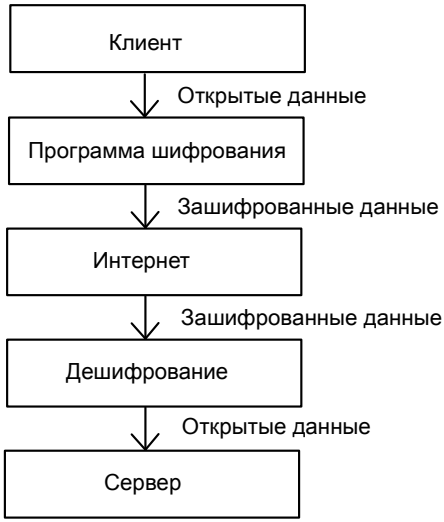


Рис. 5.4. Шифрование канала

Получается, что шифрование не изменяет протокол, он остается тем же самым TCP/IP, просто на сторонах клиента и сервера появляются посредники, которые кодируют и декодируют данные. Использование такого метода очень удобно тем, что можно шифровать трафик любого протокола. Если в криптографическом алгоритме будет найдена ошибка, или в него будут внесены изменения для более качественного шифрования (например, с использованием более длинного ключа), то не надо вносить изменения в протокол. Достаточно обновить программу шифрования, и все новые возможности станут доступными.

Рассмотрим пример с WEB-сервисом, который работает на порту 80. На сервере можно запустить программу шифрования, например на порту 1080, и все данные, которые будут приходить на него, будут декодироваться и передаваться на порт 80. Если вы хотите предоставлять доступ к WEB только по протоколу SSL, то к порту 80 можно закрыть доступ извне при помощи сете-

вого экрана (о работе сетевого экрана мы подробно говорили в *главе 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 среди пользователей Windows браузере Internet Explorer это пиктограмма висячего замка, а в некоторых версиях браузера FireFox строка адреса окрашивается в какой-то из оттенков желтого (рис. 5.5). Я не дальтоник, просто не знаю, как назвать эту неожиданность.

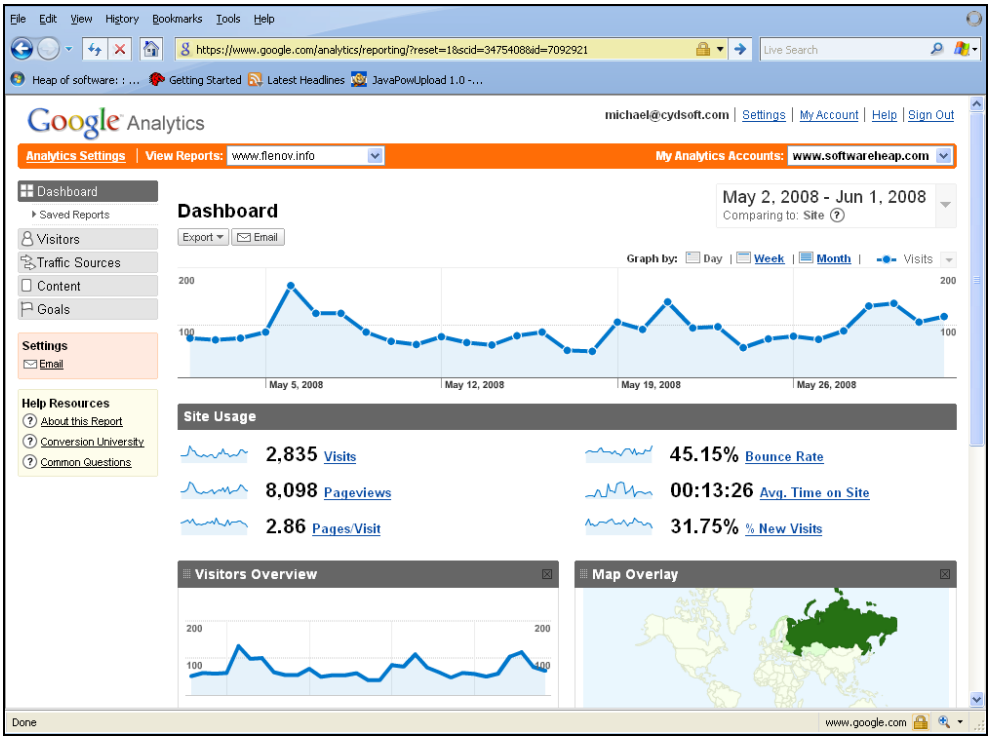


Рис. 5.5. Безопасное соединение в Firefox

Но этот опознавательный значок в Internet Explorer появляется не всегда. Более точно определить тип используемого соединения можно по свойствам документа. Большинство браузеров имеют команду просмотра свойств загруженной страницы, вызвав которую 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.

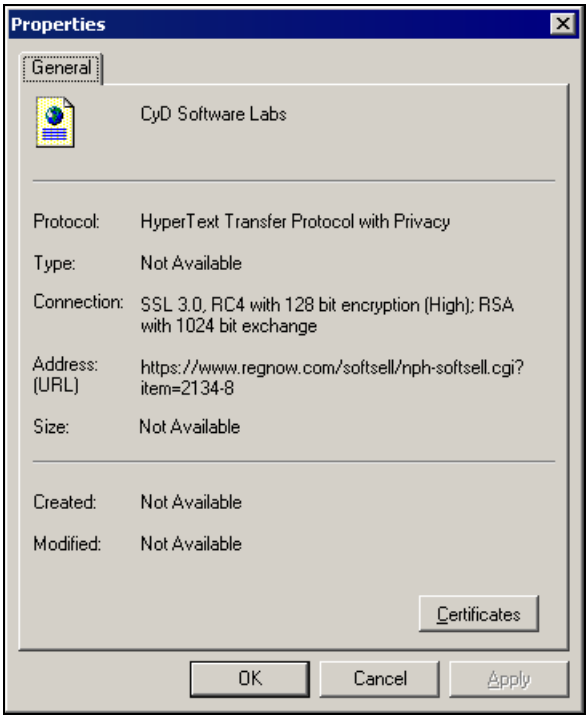


Рис. 5.6. Свойства документа в Internet Explorer

5.2.1. *stunnel*

В ОС Linux для шифрования и дешифрования трафика чаще всего используется программа stunnel. Однако она лишь организует канал и выступает посредником, а для самого кодирования используется пакет OpenSSL, доступный в большинстве дистрибутивов Linux. Вероятно, он у вас уже установлен, а если нет, то его легко установить с помощью 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, и он

посылается клиентом на сервер по открытому каналу (отсюда и его название). Сервер использует его для кодирования, но для расшифровки необходим закрытый ключ, который известен только клиенту.

У программ OpenSSL и stunnel очень много параметров, и рассматривать их все бессмысленно, потому что запомнить все проблематично, да и обычно не нужно. 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, который по умолчанию уже создан во время установки ОС или программы stunnel и 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, как в нашем примере, то прослушиваться будут все интерфейсы локального компьютера. В качестве порта был выбран номер 9002. Все пришедшие на него данные считаются зашифрованными, и они будут декодироваться для передачи на другой порт локального компьютера;
- `-r` — после этого ключа указывается имя компьютера (необязательный параметр) и порт, куда нужно передавать расшифрованные данные. Если хост не указан, то данные будут пересылаться на порт локального компьютера, в данном случае на порт 25, где должен работать SMTP-сервер. Если данные предназначены другому компьютеру, то в качестве параметра укажите `192.169.77.1:25`, где `192.169.77.1` — это IP-адрес компьютера. Однако помните, что при этом незашифрованные данные будут пересылаться по сети, чего мы как раз пытались избежать.

Теперь рассмотрим запуск клиента. Тут все намного проще:

```
stunnel -c -d 1000 -r 192.168.77.1:9002
```

Здесь у нас появился 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 (значе-

ние 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.

Если на вашей системе включен сетевой экран и запрещены подключения по указанным портам, то соединения будут отклонены сетевым экраном.

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 используется ключ `-v`, после которого идет число:

- `0` — нет никакой проверки;
- `1` — если сертификат присутствует, то он проверяется на подлинность. Если получен отрицательный результат, то соединение разрывается. Наличие сертификата не является обязательным, соединение может быть установлено и без него;
- `2` — сертификат является обязательным. Если сертификата нет, или он не является подлинным, соединение не может быть установлено;
- `3` — наличие сертификата обязательно, и помимо этого он должен присутствовать в локальном хранилище (специальный список). В этом случае нужно указать директорию, в которой находятся сертификаты, с помощью опции `-a`.

Сервер SSL может расшифровывать трафик и передавать его на порт принимающей программы не только локального, но и другого компьютера. Таким образом, сервер SSL и сервер-получатель трафика могут 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 `-T`.

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 буквы s, которая и указывает на безопасное соединение через SSL. Эта информация приведена в табл. 5.1.

Таблица 5.1. *Список протоколов с номерами портов*

Протокол	Порт TCP при обычном соединении	Название SSL-варианта протокола	Порт TCP при SSL-соединении
HTTP	80	HTTPS	443
SMTP	25	SMTPS	465
LDAP	329	LDAPS	636
TELNET	23	TELNETS	992
SHELL	514	SSH	22
FTP	21	FTPS	990
FTP-DATA	20	FTPS-DATA	989
IMAP	143	IMAPS	993
POP3	110	POP3S	995
IRC	194	IRCS	994

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 *главе 10*, когда будем рассматривать этот протокол.

5.2.3. Шифрование файлов

Некоторые серверы могут использоваться для хранения архивных данных, которые, несмотря на такой статус, должны быть скрыты от постороннего 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 электронную 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 предпочтение именно ему. О поддерживаемых алгоритмах можно узнать на сайте разработчиков или по команде `man openssl`.

Параметр `-in` задает входной файл, который необходимо шифровать, а после ключа `-out` указывается имя файла, куда сохраняется результат.

При дешифровании 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
```

В качестве примера зашифруем список 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 повторить ввод, дабы исключить возможные ошибки.

Выполните команду `cat /home/passwd` и убедитесь в том, что содержимое этого файла нечитаемо.

Для расшифровки файла выполните команду:

```
/usr/bin/openssl des -d -in /home/passwd -out /etc/passwd
```

Таким нехитрым способом мы можем безопасно хранить копию файла с паролями. К теме резервного копирования мы вернемся в *главе 13*, которая будет полностью посвящена этому вопросу.

5.2.4. Туннель глазами хакера

Хакеры могут использовать туннели для своих целей. Например, несколько лет назад я был подключен к Интернету по технологии ADSL (Asymmetric, Asymmetric Digital Subscriber, асимметричная цифровая абонентская линия). Это сейчас легко найти тарифы с безлимитным трафиком, а в начале 2000-х с этим были проблемы. В абонентскую плату входило всего 400 Мбайт трафика. Ну что такое 400 Мбайт? Для меня это неделя работы в экономном режиме, потому что общаться приходится много, а из почтового ящика я иногда

выкачиваю до 20 Мбайт в день. А превышение лимита обходилось достаточно дорого.

Как можно обойти ограничение и получить бесплатный трафик? Подобно большинству провайдеров, мой предоставлял 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 с этого сайта неограничен. И если установить на сервер программу туннелирования, то можно организовать соединение, как показано на рис. 5.7.

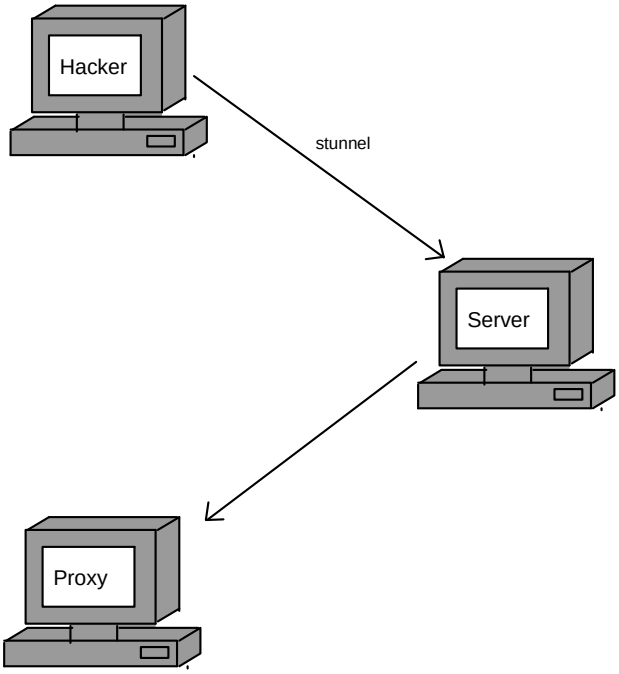


Рис. 5.7. Подключение к Интернету через WEB-сервер

Хакер подключается через программу stunnel к бесплатному WEB-серверу, который находится на площадке провайдера. Оттуда весь трафик перенаправляется на какой-нибудь прокси-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.

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 большой трафик идет 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 запрещено. 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 направляет данные на нужные порты с нужным протоколом.

Например, нам нужен доступ к FTP. На каком-нибудь сервере в Интернете на 80 порту (доступ к которому разрешен, потому что это стандартный порт для WEB-сервера) ставим сервер туннеля и настраиваем его на подключение к нужному FTP-серверу. А на своем компьютере устанавливаем клиент. С помощью клиента соединяемся 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, так как при подключении к сети Интернет без службы DNS работать сложно.

Как видите, абсолютно безопасная на первый взгляд и даже предназначенная для защиты технология превращается в средство взлома ограничений, которыми "наградил" вас администратор.

Если вы обладаете хорошими знаниями программирования на PHP или Perl, то можете написать WEB-сценарии, которые на сервере будут обращаться к необходимым вам ресурсам, чтобы работать с необходимым сервером через WEB-браузер. Получится что-то похожее на работу 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 Telnet не 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, используя при этом безопасные каналы 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 подключаясь к серверам, и нет необходимости на каждом из них держать по монитору. У меня сделано именно так (экономия на железе), и есть только один дежурный монитор, который я могу подсоединить к любому системному блоку, если надо решать проблему не по сети.

Преимущество протокола SSH заключается в том, что он позволяет удаленно выполнять команды, но при этом требует аутентификации 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 OpenSSH, для которого родной платформой можно считать другую UNIX-систему — OpenBSD — и который клонировался во все UNIX-платформы, в том числе и в Linux. Но даже сейчас в конфигурационных файлах можно иногда увидеть в комментариях имя OpenBSD.

Протокол SSH требует для работы настройки сервера и клиента. Сервер ожидает подключения и выполняет директивы пользователя, а с помощью клиента вы осуществляете соединение с сервером и посылаете запросы на выполнение команд. Таким образом, для нормальной работы вы должны правильно сконфигурировать обе части протокола.

5.3.1. Конфигурационные файлы

Все настроечные файлы протокола SSH находятся в 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 криптостойких алгоритма DSA (ssh_host_dsa_key, ssh_host_dsa_key.pub) и RSA (ssh_host_rsa_key и ssh_host_rsa_key.pub). (Замечу, что первое сокращение означает Digital Signature Algorithm или алгоритм цифровой подписи, тогда как второе, несмотря на схожесть, состоит из первых букв имен основателей одноименного алгоритма шифрования: Ronald Rivest, Adi Shamir и Leonard Adleman). Оставшиеся два файла ssh_host_key и ssh_host_key.pub хранят ключи для первой версии SSH. Для каждого алгоритма требуется по два файла: с расширением pub — хранит открытый ключ, без расширения — содержит приватный ключ.

С помощью открытого ключа можно закодировать данные и отправить их 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
# Ключ для первой версии протокола
#HostKey /etc/ssh/ssh_host_key
# HostKeys for protocol version 2
# Ключи для второй версии протокола
#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_dsa_key

# Lifetime and size of ephemeral version 1 server key
# Время жизни и размер регенерируемого серверного ключа версии 1
#KeyRegenerationInterval 3600
#ServerKeyBits 768

# Logging
# Ведение логов
#obsoletes QuietMode and FascistLogging
#SyslogFacility AUTH
SyslogFacility AUTHPRIV
#LogLevel INFO

# Authentication:
# Аутентификация

#LoginGraceTime 600
#PermitRootLogin yes
#StrictModes yes

#RSAAuthentication yes
#PubkeyAuthentication yes
#AuthorizedKeysFile       .ssh/authorized_keys

# rhosts authentication should not be used
```

```
# Аутентификацию rhosts не следует использовать
#RhostsAuthentication no
# Don't read the user's ~/.rhosts and ~/.shosts files
# Не читать принадлежащие пользователю файлы ~/.rhosts и ~/.shosts
#IgnoreRhosts yes
# For this to work you will also need host keys in
# /etc/ssh/ssh_known_hosts
# Чтобы это работало, нужны также ключи в /etc/ssh/ssh_known_hosts
#RhostsRSAAuthentication no
# similar for protocol version 2
# аналогично для версии 2 протокола
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# RhostsRSAAuthentication and HostbasedAuthentication
# Измените на yes, если не доверяете ~/.ssh/known_hosts для
# аутетификации RhostsRSAAuthentication и HostbasedAuthentication
#IgnoreUserKnownHosts no

# To disable tunneled clear text passwords, change to no here!
# Чтобы запретить туннелирование паролей, замените здесь на no
#PasswordAuthentication yes
#PermitEmptyPasswords no

# Change to no to disable s/key passwords
# Замените на no, чтобы запретить пароли s/key
#ChallengeResponseAuthentication yes

# Kerberos options
# KerberosAuthentication automatically enabled if keyfile exists
# Опции протокола Kerberos
# KerberosAuthentication разрешается автоматически, если существует файл
# с ключами
#KerberosAuthentication yes
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes

# AFSTokenPassing automatically enabled if k_hasafs() is true
# AFSTokenPassing разрешено автоматически, если вызов k_hasafs()
# возвращает true
```

```
#AFSTokenPassing yes

# Kerberos TGT Passing only works with the AFS kaserver
# Механизм билет-возврат билета в Kerberos работает только с 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
# по умолчанию баннер не задан
#Banner /some/path
#VerifyReverseMapping no

# override default of no subsystems
# перевод не особо логичен, поэтому скажу так:
# позволяет влючить sftp-сервер. Это защищенный протокол FTP с
# шифрованием. При этом отдельные демоны wu-ftp или ftpd не нужны
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 порт 80. 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 второй версии протокола, и только затем по первой. 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` — адрес для прослушивания подключения. У вашего сервера может быть несколько сетевых карт. По умолчанию идет прослушивание всех интерфейсов. Вы должны указать только те, с которых 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` — путь к файлам, содержащим ключи шифрования. Нужно прописывать только закрытые ключи, которые использует сервер для расшифровки пакетов;
- `KeyRegenerationInterval` — интервал времени, через который сервер обновляет ключи. В версии 1 во время сессии 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 первой версии протокола (см. параметр `Protocol`), то этот атрибут не влияет на работу;
- `ServerKeyBits` — длина серверного ключа. По умолчанию установлено `768`, минимальное значение — `512`;
- `SyslogFacility` — тип сообщений, которые будут сохраняться в журнале;
- `LogLevel` — уровень событий, которые будут попадать в журнал. Возможные уровни соответствуют системным, которые мы будем рассматривать в *разд. 12.5*;
- `LoginGraceTime` — интервал времени, в течение которого пользователь должен ввести правильный пароль, иначе соединение будет разорвано;
- `PermitRootLogin` — флаг, определяющий, разрешено (`yes`) или запрещено (`no`) входить в систему по протоколу SSH под пользователем root. Мы уже говорили, что root — это бог в системе, и его возможности нужно использовать аккуратно. Если в систему нельзя входить с такими правами, то и по SSH тем более. Срочно меняйте этот параметр на `no`;
- `StrictModes` — флаг, регламентирующий необходимость проверки состояния файлов и их владельцев, пользовательских файлов и домашней директории до ввода пароля. Желательно установить `yes`, потому что многие начинающие пользователи делают все свои файлы доступными для записи.

- `RSAAuthentication` — разрешена ли аутентификация по алгоритму RSA. Параметр действует для первой версии протокола;
- `PubkeyAuthentication` — дозволена ли аутентификация по публичному ключу. Параметр действует для второй версии протокола;
- `AuthorizedKeysFile` — файл с публичным ключом, который может использоваться для аутентификации;
- `RhostsAuthentication` — флаг, разрешающий аутентификацию по файлам $HOME/.rhosts и /etc/hosts.equiv. По умолчанию стоит `no`, и без особой надобности менять этот параметр не стоит, потому что это небезопасно;
- `IgnoreRhosts` — флаг, запрещающий читать файлы ~/.rhosts и ~/.shosts. Без необходимости значение лучше не изменять, потому что это может повлиять на безопасность;
- `AuthorizedKeysFile` — файл для хранения списка авторизованных ключей. Если пользователь входит в систему с имеющимся в этом файле ключом, то его пустят автоматически без ввода дополнительных паролей;
- `RhostsRSAAuthentication` — флаг, регламентирующий использование ключа хоста из директории /etc/ssh/ssh_known_hosts при аутентификации. Параметр применяется в первой версии SSH;
- `IgnoreUserKnownHosts` — параметр, указывающий на необходимость игнорирования пользовательских списков доверенных компьютеров. Если он равен `no`, то необходимо доверять компьютерам из списка ~/.ssh/known_hosts при RhostsRSAAuthentication-аутентификации. Не верьте никому, поэтому параметр лучше всего изменить на `yes`;
- `PasswordAuthentication` — требование пароля. Если значение равно `yes`, то будет требоваться пароль. При использовании авторизации через ключи параметр можно отключить;
- `PermitEmptyPasswords` — разрешение пустых паролей. По умолчанию установлено `no`, что запрещает использование пустых паролей, и изменять это значение не стоит;
- `KerberosAuthentication` — использование проверки подлинности по протоколу Kerberos. В последнее время именно эта аутентификация набирает большую популярность благодаря своей безопасности;
- `KerberosOrLocalPasswd` — если пароль Kerberos не был принят, то включается проверка локального файла паролей из файла /etc/shadow;
- `KerberosTicketCleanup` — удаление билета Kerberos из кэша при выходе из системы;
- `Banner` — позволяет указать файл, в котором находится текст приветствия, отображаемого пользователям.

5.3.3. Параметры доступа к серверу *sshd*

Кроме директив, приведенных в листинге 5.1, можно использовать следующие:

- `AllowGroups` — позволить вход в систему только пользователям указанных групп (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
# Значения некоторых параметров для всех пользователей по умолчанию

# Host *
#   ForwardAgent no
#   ForwardX11 no
#   RhostsAuthentication yes
#   RhostsRSAAuthentication yes
#   RSAAuthentication yes
#   PasswordAuthentication yes
#   FallBackToRsh no
#   UseRsh no
```

```
#   BatchMode no
#   CheckHostIP yes
#   StrictHostKeyChecking ask
#   IdentityFile ~/.ssh/identity
#   IdentityFile ~/.ssh/id_rsa
#   IdentityFile ~/.ssh/id_dsa
#   Port 22
#   Protocol 2,1
#   Cipher 3des
#   Ciphers aes128-cbc,3des-cbc,blowfish-cbc,cast128-cbc,arcfour
#   EscapeChar ~
Host *
Protocol 2,1
```

Некоторые из этих параметров мы уже видели при рассмотрении серверного конфигурационного файла. Здесь также можно увидеть параметр `Protocol`, в котором указываются используемые версии SSH. В данном случае не стоит запрещать версию 1. На безопасность клиента это не повлияет, зато не будет проблем при подключении к серверу, который работает только на такой версии. Я надеюсь, что это будет не ваш сервер :).

Вот характерные для клиента команды:

- `Host` — сервер, к которому относятся следующие настройки;
- `CheckHostIP` — разрешение проверки IP-адреса с перечисленными в файле known_hosts адресами;
- `Compression` — разрешение (`yes`) или запрет (`no`) использования сжатия данных;
- `KerberosAuthentication` — разрешение (`yes`) или запрет (`no`) использования аутентификации по протоколу Kerberos;
- `NumberOfPasswordPrompts` — количество попыток ввода пароля. Если пароль не введен верно, то соединение разрывается;
- `IdentityFile` — имя файла, содержащего закрытые ключи пользователя;
- `PasswordAutentication` — режим 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 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.

Подключаться к SSH-порту Linux вы можете и из Windows. Для этого существует множество различных клиентов.

5.3.6. Вход по ключу

Намного удобнее и безопаснее способ авторизации по ключу, а вход по паролю может быть даже и вовсе заблокирован. Обращение к системе по SSH не вполне безопасно. Злоумышленник может подсмотреть пароль, когда вы будете вводить его в другой программе. Тогда зачем шифровать SSH-соединение, если секретное слово может быть выявлено при работе с другими программами?

Для каждого подключения должны быть свои пароли. Но помнить их все очень сложно, поэтому лучше для авторизации использовать ключи, которые и так защищены дальше некуда. Нужно сделать только небольшие изменения в конфигурации.

Для начала нужно создать новый ключ. Для этого используется команда `ssh-keygen`. У нее есть такие параметры:

- `-t` — тип ключа. Здесь можно указывать `rsa` или `dsa` для второй версии SSH или `rsa1` — для первой. Для примера будем использовать ключ `rsa`;

- `-f` — файл, в котором будет сохранен закрытый ключ. Открытый ключ получит такое же имя, но с расширением pub;
- `-b` — длина ключа, которая может быть не меньше `512`. По умолчанию установлено значение `1024`, оставим его и не будем указывать этот параметр.

Итак, для генерации ключа выполним команду:

```
ssh-keygen -t rsa -f ~/.ssh/myrsakey
```

Обратите внимание, что я указал сохранение ключа в поддиректории .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 его имя будет id_dsa. Я специально задал имя, чтобы показать, как с ним работать.

Если программа запустилась успешно, то на экране вы должны увидеть следующее приглашение:

```
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
```

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 целое предложение). После нажатия клавиши <Enter> вам предложат подтвердить комбинацию, чтобы исключить ошибки при вводе.

Если все прошло успешно, то вы должны увидеть следующее сообщение:

```
Your identification has been saved in ~/.ssh/myrsakey.
Your public key has been saved in ~/.ssh/myrsakey.pub.
```

В первой строке нас проинформировали о том, что закрытый ключ сохранен в файле ~/.ssh/myrsakey, а во второй — что открытый сохранен 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` — разрешение переправлять 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. Если параметр равен `yes`, то 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 клиент X-Win32, который можно скачать с сайта **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 SSH-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 протокола FTP. Этот протокол мы будем подробно рассматривать в *главе 10*, а сейчас вам достаточно знать, что большинство команд схожи с директивами Linux для управления файлами.

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 протокола FTP 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 поддерживают шифрование SSH, поэтому убедитесь в поддержке этого протокола со стороны вашего программного обеспечения. Например, если вы работаете в Windows

и хотите использовать безопасное соединение с Linux для обновления файлов, можете воспользоваться программой WinSCP (**www.winscp.net**). Она обладает простым и удобным графическим интерфейсом, а любители движения OpenSource смогут даже найти исходные коды программы.

5.4. Демон *inetd/xinetd*

Для того чтобы сервер смог обрабатывать запросы клиентов, программа должна быть постоянно загружена и связана с определенным портом. В этом нет ничего сложного, но глупо постоянно держать программу в памяти, если она большая, а работает очень редко. В ряде случаев лучше, когда один сервис в системе будет следить за портами и запускать необходимый сервис при обращении к определенному каналу. В ОС Linux такая возможность есть, и 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 протоколами TCP и UDP, порты которых не пересекаются (они совершенно различны), так что необходимо в явном виде указывать протокол;
- `Псевдоним` — вымышленное имя для сервиса, которое можно не указывать.

Например, в файле /etc/services вы найдете следующие строки:

```
tcpmux          1/tcp           # TCP port service multiplexer
tcpmux          1/udp           # TCP port service multiplexer
rje             5/tcp           # Remote Job Entry
rje             5/udp           # Remote Job Entry
echo            7/tcp
...
...
ftp             21/tcp
ftp             21/udp              fsp fspd
```

Я специально выбрал эти строки, чтобы вы увидели варианты описания различных сервисов.

Если вы используете старый дистрибутив ОС Linux, то в нем, скорей всего, еще работает init.d. Но мы уже говорили, что давнишний дистрибутив не может

быть безопасным, и с ним что-либо делать бесполезно. Лучшая защита — обновление, и тогда основным сервисом станет xinetd, который становится, если еще не стал, стандартом для всех.

Я рекомендую переход на xinetd, потому что в нем появилось много дополнительных возможностей, которые повышают удобство администрирования и безопасность. Например, в 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, а также директория, в которой будут находиться конфигурационные файлы, влияющие на работу конкретных сервисов. Рассмотрим приведенное в листинге 5.3 содержимое этого файла.

Листинг 5.3. Файл конфигурации /etc/xinetd.conf

```
#
# Simple configuration file for xinetd
# Пример конфигурационного файла для xinetd
#
# Some defaults, and include /etc/xinetd.d/
# Некоторые параметры по умолчанию
# и подключение 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. В этом каталоге для каждой службы есть собственный конфигурационный файл, где можно изменить параметры. Имена файлов соответствуют названиям сервисов, а содержимое — похоже на /etc/xinetd.conf. В листинге 5.4 приведено содержимое конфигурационного файла /etc/xinet.d/telnet для сервиса Telnet.

Листинг 5.4. Конфигурационный файл для сервиса Telnet

```
# default: on
# По умолчанию включен
# description: The telnet server serves telnet sessions; it uses \
#       unencrypted username/password pairs for authentication.
# Описание: telnet-сервис обслуживает сессии telnet. Он использует
# незашифрованные имя пользователя и пароль для аутентификации
service telnet
{
       disable            = no
       flags              = REUSE
       socket_type        = stream
       wait               = no
       user               = root
       server             = /usr/sbin/in.telnetd
       log_on_failure    += USERID
}
```

Рассмотрим основные параметры, которые можно изменять:

- `disable` — параметр, установка которого в `yes` запрещает исполнение сервиса. Как уже говорилось, сервис telnet небезопасен, и поэтому имеет смысл его запретить;
- `flags` — атрибуты выполнения сервиса;
- `socket_type` — тип используемого сокета. Для протокола TCP здесь должно быть значение `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` — файл или системный журнал для записи событий;
- `log_on_success` и `log_on_failure` — информация, которая будет сохраняться в журнале при удачном и неудачном входе в систему соответственно. Здесь можно указывать значения: `PID`, `HOST` или `USER`;
- `per_source` — максимальное количество соединений от одного пользователя. Их может быть несколько, потому что пользователи любят максимально нагружать каналы и повышать скорость работы с помощью создания нескольких одновременно работающих соединений;
- `server_args` — аргументы, с которыми будет запускаться сервер.

При рассмотрении параметра `user` я упомянул о необходимости прав администратора для доступа к портам с номером менее 1024. Это действительно так, но зачем это нужно? 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, причем, возможно, ему удалось бы подменить конфигурационный файл при запуске, чтобы расширить свои права. Сделай он это, и у него появится возможность загружать на сервер файлы и скачивать их к себе на компьютер, что нежелательно.

5.4.2. Безопасность

Мы уже знаем, что программа xinetd позволяет определять права и время доступа к сервисам. Для этого в конфигурационном файле можно использовать три директивы: `no_access`, `only_from` и `access_time`.

Директива `no_access` запрещает доступ с указанных компьютеров. Например, следующая строка в конфигурационном файле закрывает доступ с адреса 192.168.1.1:

```
no_access 192.168.1.1
```

Если необходимо запретить доступ целой сети, то достаточно указать только ее номер. Например, если нужно закрыть вход всем компьютерам с адресами 192.168.1.X, где X может быть любым числом, следует использовать следующую строку:

```
no_access 192.168.1.
```

Обратите внимание, что в этом случае указанный IP-адрес состоит из трех октетов, разделенных точками, а не четырех, правда после третьего числа указывается точка.

Для полного запрета доступа необходимо указать в конфигурационном файле следующую строку:

```
no_access 0.0.0.0
```

Теперь посмотрим, как можно предоставлять доступ с помощью директивы `only_from`. Она удобна тем, что изначально запрещает все, кроме указанных в качестве параметра адресов. Получается, что мы действуем от запрета. Указав эту команду без параметра, мы вообще запретим доступ:

```
only_from =
```

Я рекомендую включить эту строку в основной конфигурационный файл /etc/xinetd.conf, а потом для каждого отдельного сервиса в его собственном конфигурационном файле прописать разрешения. Например, давайте откроем доступ с адресов 192.168.1.2 и 192.168.1.19. Для этого добавляем в конфигурационный файл следующую строку:

```
only_from = 192.168.1.2 192.168.1.19
```

Можно разрешать доступ целым сетям:

```
only_from = 192.168.1.
```

А что, если всей сети разрешен доступ, а компьютеру с номером 1 запрещен? В этом случае можно 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 сотрудники часто задерживаются на работе, и не хочется, чтобы они дергали меня каждый день из-за доступа.

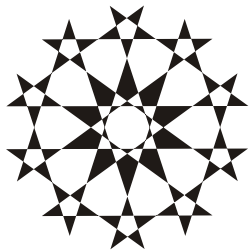
Функции обеспечения безопасности, встроенные в сервис xinetd, удобны и обладают достаточно мощными возможностями, хотя и дублируют права доступа, которые можно настраивать из файлов /etc/hosts.allow и /etc/hosts.deny.

Мне больше нравится заниматься настройкой безопасности с помощью конфигурационных файлов сервиса xinetd, потому что параметры доступа хранятся в тех файлах, на которые они влияют.

5.4.3. Недостатки *xinetd*

У любой технологии есть недостатки, и inetd/xinetd не являются исключением. При подключении пользователя к одному из портов вашего сервера программа inetd (или xinetd) 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 inetd (или xinetd) съест все ресурсы компьютера 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 протокол FTP (об этом мы подробно поговорим в *главе 10*). Но он неудобен, так как использует технологию "клиент-сервер". Чтобы вы могли получить с компьютера друга файл, он должен запустить у себя FTP-сервер, а вы с помощью специальной программы, называемой FTP-клиентом, должны подключиться к этому серверу и скачать нужный файл. Сложность заключается не только в необходимости установки и настройки различных программ, но и в контроле доступа.

Чтобы не утруждать себя настройками FTP-сервера, многие стали использовать специализированные обменники в своих локальных сетях. Для этого выделялся FTP-сервер с открытым доступом, который быстро превращался в мусорную корзину.

В Windows появился более удобный способ публиковать свои файлы — "Сетевое окружение", с помощью которого удаленный пользователь может зайти на любой компьютер и использовать его открытые ресурсы. Это действительно хорошая возможность, и пользователи привыкли к ней, несмотря на то, что работа с открытыми ресурсами была небезопасной.

Чтобы пользователи Windows могли видеть сервер Linux в своем сетевом окружении и работать с ним как с Windows-машиной, был разработан пакет Samba (часто можно встретить сокращение smb), который состоит из двух программ. Сервер позволяет публиковать локальные папки для всеобщего просмотра, а с помощью клиента вы можете подключаться к другим компьютерам и работать с их открытыми ресурсами.

С помощью Samba можно сделать легкодоступный и удобный в использовании файловый сервер. Раньше у меня на работе для этих целей использовался сервер Windows 2000, который параллельно работал как сервер баз данных.

Но файловый архив занимает слишком много места, отнимает ресурсы сети и сервера, а также понижает безопасность. К тому же, использовать дорогостоящую ОС Windows как файловое хранилище просто глупо.

Поэтому было принято решение перенести файловый архив на отдельный физический сервер с бесплатной ОС. 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. Ну зачем графический интерфейс файловому хранилищу? Для этих целей лучше подойдет Linux, который обходится дешевле, если вы знаете его и умеете использовать. 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 сопоставление IP-адресов и имен компьютеров аналогично /etc/hosts в Linux и Диск:\Windows\System32\drivers\etc\lmhosts.sam в Windows.

Дополнительно в этой же директории можно создать следующие файлы (некоторые из них могут существовать):

- smbusers — список пользователей, которым разрешено подключаться к серверу Samba;
- smbpasswd — пароли пользователей из файла smbusers;

Как видите, у Samba свои конфигурационные файлы для хранения списка пользователей. Если вы создаете их вручную, то убедитесь, что права на чтение и запись установлены правильно. Файл должен быть доступен только администратору, то есть владельцем может быть только root и никто иной.

Конфигурационный файл smb.conf содержит не так много директив, поэтому для удобства восприятия я привел в листинге 6.1 небольшой пример, который поможет вам увидеть общую структуру такого файла. В дальнейшем нам предстоит рассматривать другие серверы Linux, где настроек намного больше.

Листинг 6.1. Фрагмент конфигурационного файла smb.conf

```
 [global]
# Основные директивы
   workgroup = MYGROUP
   server string = Samba Server

;   hosts allow = 192.168.1. 192.168.2. 127.
   load printers = yes
   printing = lprng
;   guest account = pcguest

# Директивы журнала
   log file = /var/log/samba/%m.log
   max log size = 0
   syslog only = no

# Директивы безопасности
   security = user
;   password server = <NT-Server-Name>
;   password level = 8
;   username level = 8
   encrypt passwords = yes
   smb passwd file = /etc/samba/smbpasswd
;   usershare max shares = 100

   unix password sync = Yes
   passwd program = /usr/bin/passwd %u
   passwd chat = *New*password* %n\n *Retype*new*password* %n\n
   *passwd:*all*authentication*tokens*updated*successfully*
   pam password change = yes
;   username map = /etc/samba/smbusers

;   include = /etc/samba/smb.conf.%m
   obey pam restrictions = yes

# Настройка сокета
   socket options = TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
;   interfaces = 192.168.12.2/24 192.168.13.2/24
```

```
;   bind interfaces only = yes

# Настройка просмотра
;   remote browse sync = 192.168.3.25 192.168.5.255
;   remote announce = 192.168.1.255 192.168.2.44
;   local master = no
;   os level = 33
;   domain master = yes
;   preferred master = yes

# Работа с сервером
;   domain logons = yes
;   logon script = %m.bat
;   logon script = %U.bat
;   logon path = \\%L\Profiles\%U
;   wins support = yes

# WINS сервер
   wins support = no
   wins server = w.x.y.z
   dns proxy = no
   name resolve order = lmhosts host wins bcast

# Отображение файлов
;   preserve case = no
;   short preserve case = no
;   default case = lower
;   case sensitive = no
```

Реальный файл в вашей системе будет намного больше, потому что он содержит множество комментариев с описаниями и примерами конфигурирования открытых 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, приближенный к файлам настроек Windows:

```
ИмяДирективы=Значение
```

Значение директивы ставится после знака равенства. Таким образом, имя директивы может состоять из нескольких слов, содержать пробелы и различные символы (кроме знака равенства). Строки, начинающиеся с точки с запятой (`;`) или с диеза (`#`), рассматриваются как комментарии.

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` — 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` — имя, которое пользователи будут видеть в сетевом окружении для данного сервера, оно не должно совпадать с именем рабочей группы;
- `server string = описание` — свободный текст, который можно будет увидеть в поле **Description** (Комментарий) свойств сервера или окне сетевого окружения в режиме **Details** (Таблица). В этом поле вы можете поместить комментарий, описывающий содержимое сервера, например, "Файловый архив Сергея";
- `hosts allow = адреса` — 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 с адресом 192.168.2.2 из другой сети надо написать следующую строку:

  ```
  hosts allow = 192.168.1. 192.168.2.2
  ```

- `printcap name = файл` — файл описания принтеров, подключенных к системе. По умолчанию это /etc/printcap;
- `load printers = yes` — установка режима (yes) автоматического включения принтеров в список открытых ресурсов. Если в этом нет надобности, то укажите `no`;
- `printing = система` — тип системы печати. Здесь можно использовать одно из следующих значений: `bsd`, `sysv`, `plp`, `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` — указывает учетную запись, с правами которой пользователь сможет входить в систему. Если ваш сервер не содержит секретной информации и используется для открытого обмена файлами, то можно завести гостевую учетную запись, иначе такой вход не безопасен;
- `log file = файл` — задает название файла-журнала, например, можно написать /var/log/samba/%m.log. Обратите внимание, что в имени присутствует символ процента, за которым следует буква m. Эта комбинация во время работы будет заменена именем пользователя, активность которого сохраняется. Так, например, для пользователя robert будет создан журнал /var/log/samba/robert.log;
- `max log size = n` — определяет максимальный размер журнала в килобайтах. Укажите значение `0`, если ограничения не должно быть;
- `syslog only` — инструктирует использовать только syslog для журналирования;
- `security = уровень` — определяет степень доступа. Параметр `уровень` может принимать одно из следующих значений:
  - `user` — доступ на уровне пользователя;
  - `share` — аутентификация на основе имени и пароля;
  - `server` — проверка пароля производится сторонним smb-сервером. Имя сервера, на котором хранятся пароли, задается с помощью директивы `password server = ИмяСервера`, что позволяет держать пароли на другом smb-сервере, который и будет осуществлять проверку пароля;
  - `domain` — включает сервер в домен Windows NT, а пароль для доступа указывается в файле, определенном с помощью директивы `smb passwd file = ИмяФайла`;
- `encrypt passwords = yes` — позволяет шифровать пароли, передаваемые по сети. Данная опция требует пояснений, потому что могут возникнуть проблемы при авторизации с компьютеров с системой Windows.

  Суть в том, что Windows 95 вообще не шифровал пароли, и они передавались в открытом виде. Начиная с Windows 98 по умолчанию система стала шифровать пароли. Зашифрованный пароль передается по сети, и на сервере происходит дешифровка. Если система ожидает зашифрованного пароля, а клиент посылает незашифрованный, то с соединением возникают серьезные проблемы.

Современные версии Windows шифруют пароль, но вы можете отключить эту возможность. Для этого в реестре нужно изменить параметр `EnablePlainTextPassword`, установив в нем значение `1`. В Windows 9*x* этот параметр находится в реестре по адресу:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\VxD\VNETSUP

Для Windows NT это:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Rdr\Parameters

В Windows 2000 и более поздних версиях:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanWorkstation\Parameters

Если соответствующего параметра не существует, его следует создать. Он должен иметь тип Dword.

Если возникли проблемы с входом в систему, то переключите систему в режим работы с незашифрованными паролями. В этом случае Samba использует для авторизации файлы /etc/passwd и /etc/shadow. Полученный открытый пароль шифруется по алгоритму MD5 и сравнивается со значениями из этого файла.

Если вы используете шифрованный пароль, то при авторизации будет использоваться файл /etc/samba/smbpasswd (это можно изменить с помощью директивы `smb passwd file`). Этот файл 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 расположение файла с паролями. По умолчанию он находится в той же директории, где расположены конфигурационные файлы Samba;
- `ssl CA certFile = файл` — задает файл сертификата, необходимый для работы протокола SSL, гарантирующего безопасность передачи данных;
- `unix password sync = yes` — разрешает пользователям Windows менять пароль Samba, одновременно обновляя системные пароли в Linux. Если в этом нет необходимости, установите значение параметра `no`. Для работы этой директивы нужно указать программы, которые будут изменять паро-

ли (параметр `passwd program`) и сообщения, появляющиеся перед пользователем (параметр `passwd chat`). Приведу пример использования:

```
unix password sync = Yes
passwd program = /usr/bin/passwd %u
passwd chat = *New*password* %n\n *Retype*new*password* %n\n
*passwd:*all*authentication*tokens*updated*successfully*
```

Помимо этого, необходимо использовать директивы `encrypt passwords` и `smb passwd file`;

- `username map = файл` — указание на файл, где хранится список пользователей сервера Samba для клиентов Windows 9*x* (о нем подробнее мы поговорим в *разд. 6.3*);
- `usershare max shares` — максимальное количество пользовательских директорий, которые могут быть открыты для общего доступа;
- `usershare allow guests = yes` — если этот параметр включен, то гости системы смогут получить доступ к открытым ресурсам. По умолчанию этот параметр отключен, чтобы доступ к открытым ресурсам получали только авторизованные пользователи.

6.1.3. Сеть

В этом разделе мы изучим директивы настройки сетевого протокола:

- `include = файл` — позволяет подключить дополнительный конфигурационный файл, например, с другого компьютера, или содержащий настройки для конкретного подключающегося компьютера. В последнем случае имя файла обычно задается в формате `путь.%m`. В данном случае `путь` — это начало полного имени файла, 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 (без задержек) передавать данные;
  - `SO_RCVBUF` — размер принимающего буфера;
  - `SO_SNDBUF` — размер передающего буфера;
- `interfaces = интерфейсы` — если у вас установлены две сетевые карты, которые направлены на разные сети, то с помощью этой директивы можно позволить работать с Samba пользователям из какой-то конкретной или из обеих сетей;

- `bind interfaces only` — позволяет привязаться только к именованным интерфейсам или сетям, перечисленным в параметре `interfaces`. Если ваша сеть не защищена сетевым экраном, то настоятельно рекомендуется использовать этот параметр, чтобы доступ к серверу Samba имели только компьютеры из вашей сети, а не любой удаленный пользователь из Интернета.

6.1.4. Замена сервера Windows

Если посмотреть на параметры, которые мы рассматриваем в этом разделе, можно увидеть, что Samba способна полностью заменить Windows-сервер, и рабочие станции на базе Windows не заметят неудобств:

- `local master = yes` — позволяет сделать Samba-сервер основным браузером сети;
- `domain master = yes` — дает возможность сделать Samba-сервер главным браузером домена. Не устанавливайте значение `yes`, если в вашей сети уже есть контроллер домена Windows NT;
- `domain logons = yes` — позволяет серверу Samba работать как серверу для входа в систему компьютеров с ОС Windows 95. При загрузке компьютера с Windows можно будет вводить пароли, хранящиеся в Samba;
- `logon script = файл` — если директива `domain logons` включена, то в этом параметре можно указать файл bat-сценария, который будет выполняться на компьютере клиента при входе в систему. Сценарий может, например, задаваться в виде `%m.bat` (заменяется именем компьютера) или `%u.bat` (подменяется на имя пользователя);
- `logon path = путь` — определяет место хранения пользовательских профилей. При использовании этой директивы необходимо убрать комментарии с секции `[Profiles]`, которую можно найти в файле конфигурации по умолчанию.

6.1.5. Поддержка WINS и DNS

WINS (Windows Internet Naming Service, служба имен Интернет для Windows) — это сервис для сопоставления имен компьютеров и IP-адресов. База данных WINS чем-то напоминает DNS (Domain Name Service), только в ней хранятся NetBIOS-имена компьютеров, а DNS обслуживает доменные имена.

Для настройки работы через WINS используются следующие директивы:

- `wins support = yes` — разрешить использование WINS-сервера;
- `wins server = w.x.y.z` — IP-адрес WINS-сервера;

- `DNS Proxy = yes` — если эта опция разрешена, то не распознанные NetBIOS-имена можно попытаться определить через DNS-сервер;
- `name resolve order` — последовательность, в которой будет производиться поиск адреса хоста по его имени. В конфигурационном файле можно найти следующий пример:

  ```
  name resolve order = lmhosts host wins bcast
  ```

  Если использовать это значение, то при поиске адреса система сначала будет искать имя хоста в файле lmhosts, потом в hosts, потом будет задействован wins, и последним будет использоваться bcast (broadcast или широковещательный запрос).

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` — не принимать во внимание различие в регистре;
- `default case = lower` — отображать все файлы в нижнем регистре;
- `preserve case = no` и `short preserve case = no` — запретить сохранение имен файлов с учетом регистра.

Если у вас в сети работают пользователи Windows-систем, то рекомендуется оставить указанные 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. Это делается в отдельных секциях, которые идут после секции `[Global]`, рассмотренной в *разд. 6.1*.

6.2.1. Пора домой

Конечно же, любой пользователь захочет работать со 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 ; %S - строка с перечислением пользователей
   create mode = 0664
   directory mode = 0775
```

В этой секции могут присутствовать следующие директивы:

- `comment` — текстовый комментарий, который не влияет на работу;
- `browseable = no` — режим отображения ресурса в списке просмотра. Если указать `yes`, то в сетевом окружении будут видны папки пользователей;
- `writable = yes` — предикат записи в 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 — бесправные. Иногда следует понизить значение параметра до 740, чтобы пользователи группы могли только 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 им 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` — 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 сервер домена, то необходимо убрать комментарии с секции `[netlogon]`:

```
; [netlogon]
;   comment = Network Logon Service
```

```
;   path = /usr/local/samba/lib/netlogon
;   guest ok = yes
;   writable = no
```

В этой секции также присутствуют комментарий и директива `writable`. Пользователи не должны иметь право записи в эту директорию, потому что здесь хранятся сценарии, которые выполняются при входе в систему. Файлы из этого каталога должен изменять только администратор.

Помимо этого, есть еще две команды:

- `path = путь` — полный путь к директории netlogon;
- `guest ok = yes` — разрешение гостевого входа.

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 принтеры с общим доступом. Вроде безобидно, а работу тормозит, и бесполезно расходуется картридж.

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 файлы и читать 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.

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. Однако 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, из-за которой Samba работает неверно.

Но вы можете создавать собственные разделы, в которых будете описывать права. Например, пусть вам нужно организовать общую директорию, в которой файлы будут доступны всем пользователям, но записывать туда смогут только пользователи определенной группы. Допустим, что в этом каталоге должны храниться какие-то картинки. Для решения 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 файлы. Для всех остальных директория открыта только для чтения.

В таких объявлениях директорий можно использовать любые директивы, которые мы рассматривали в данной главе.

6.2.6. CD-ROM

Отдельным разделом в файле конфигурации идут настройки прав доступа к CD-ROM. Пример этих настроек показан ниже:

```
;[cdrom]
;   comment = Samba server's CD-ROM
;   read only = yes
;   path = /cdrom
;   guest ok = yes
```

Обратите внимание, что в начале каждой строки стоит символ точки с запятой. Я не случайно оставил их на месте, чтобы указать, что по умолчанию они есть (по крайней мере у меня). Если у вас их нет, то срочно подумайте о том, чтобы проштудировать всю систему очень подробно, мало ли что еще ваш производитель разрешил по умолчанию.

Итак, все отключено, а значит компакт-диск не будет виден по сети. Если нужно открыть к нему доступ, настройте следующие параметры:

- `comment = Samba server's CD-ROM` — комментарий;
- `read only = yes` — файлы доступны только для чтения;
- `path = /cdrom` — путь к диску. Если у вас два и более дисков, нужно указать, какой из них вы хотите расшарить, если же только один, то все равно нужно указать путь, куда он смонтирован;
- `guest ok = yes` — разрешить ли доступ к диску для гостей (неавторизованных в системе пользователей).

6.3. Управление пользователями

Для начала разберемся с именами пользователей. По умолчанию для доступа к серверу Samba используются имена из системного файла /etc/passwd. Но вы можете завести отдельные записи Samba-сервера, которые будут соответствовать реальным, но их можно будет использовать только для подключения к Samba, а не к системе.

Имена Samba описываются в файле /etc/samba/smbusers, расположение и название которого могут быть изменены директивой `username map` файла smb.conf. Содержимое файла /etc/samba/smbusers может быть, например, таким:

```
# Unix_name = SMB_name1 SMB_name2
root = administrator admin
nobody = guest pcguest smbguest
```

У списка имен несколько предназначений. Во-первых, с его помощью вы можете отразить имена, привычные для пользователей DOS и Windows, на учетные записи Linux. Например, в Windows максимальные права принадлежат пользователю Administrator, а в Linux это root. Во второй строке приведенного выше примера устанавливается соответствие имени Administrator пользователю root.

Во-вторых, использование этого файла позволяет аккумулировать несколько имен на одной учетной записи. Например, у вас есть группа пользователей, которым должны быть назначены одинаковые права. Для этого заводим в Linux только одну учетную запись `nobody` (под ней будут работать пользователи), а в smb будет несколько пользователей `guest`, `pcguest` и `smbguest` (под этими именами они будут входить в систему).

Несмотря на то, что мы отразили имена пользователей administrator и admin, и это разные учетные записи, для них будет использоваться один пароль — назначенный пользователю root.

Информация о пользователях, которым разрешен доступ, хранится в файле /etc/samba/smbpasswd. Его расположение и имя могут быть изменены с помощью директивы `smb passwd file` файла smb.conf. Рассмотрим пример содержимого файла:

```
flenov:0:813D6593C11F1173ED98178CA975D79:[UX      ]:LCT-41FA818F
robert:500:813D6593C11F1173ED98178CA975D79:[UX      ]:LCT-41FA818F
```

Сразу видно, что файл чем-то похож на /etc/passwd. Он также разделен двоеточиями на несколько колонок. Наиболее интересны из них первые три — имя пользователя, его UID в системе Linux и пароль.

Но добавлять пользователей вручную не очень удобно, потому что нужно зашифровать и прописать пароль, что не так уж и просто. Чтобы облегчить задачу, в пакет Samba включена утилита `smbpasswd`, которая принимает следующие параметры:

- `-a` — добавить пользователя в систему. Учетная запись должна уже существовать в /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, и версии Windows 9*x* делают это небезопасным способом. Если злоумышленник сможет украсть пароль на Samba, то он проникнет и в систему;
- `-x` — удалить пользователя. Чтобы исключить Роберта из системы, выполните команду `smbpasswd -x robert`;
- `-d` — деактивировать пользователя. Если необходимо временно отключить доступ для пользователя, не удаляя его из системы, выполните команду `smbpasswd -d robert`. Давайте посмотрим на строку, соответствующую Роберту, после выполнения этой команды:

  ```
  robert:500:813D6593C11F1173ED98178CA975D79:[DUX  ]:LCT-41FA818F
  ```

  Обратите внимание, что в четвертой колонке в квадратных скобках появилась буква D. Она как раз и указывает на то, что запись деактивирована. Таким образом вы легко можете определить, какие записи активны, а какие нет;
- `-e` — активировать пользователя. С помощью этой команды можно подключить деактивированного ранее пользователя: `smbpasswd -e robert`.

Дополнительные параметры этой утилиты можно посмотреть, используя справочную систему 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 доступа в систему. К таким пользователям относятся системные учетные записи типа bin, adm, deamon и др.

6.4. Использование Samba

Сервис Samba в основном создавался для пользователей Windows, но и поклонники Linux тоже оценили все преимущества этой технологии, тем более что ОС Linux управляет совместным доступом к файлам по сети не хуже Windows, а в чем-то даже лучше. Для работы с Samba из ОС Linux используется команда `smbclient`.

Чтобы подключиться к серверу необходимо указать как минимум два ключа: `-L` (адрес сервера) и `-U` (имя пользователя). В ответ на это программа запросит ввести пароль. Если вы не используете шифрование, то необходимо ввести системный пароль, в противном случае воспользуйтесь тем, который вы указали при переносе пользователя в файл /etc/samba/smbpasswd (при запуске команды `smbpasswd`).

Итак, выполните следующую команду для тестирования сервера:

```
smbclient -L localhost -U root
```

После ввода пароля пользователя root вы должны увидеть все открытые ресурсы сервера. Результат выглядит примерно 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 директорий из раздела `[homes]` файла конфигурации директива `browseable` установлена в значение `no` (*см. разд. 6.2.1*). Это значит, что таких каталогов не будет видно. Это вполне логично, потому что злоумышленнику нельзя 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 подать команду `smbclient`, передав ей имя ресурса, которое задается в формате UNC (Universal Naming Convention, универсальное именование объектов) с применением следующего синтаксиса:

```
\\ИмяСервера\ресурс
```

Например, вы хотите подключить домашнюю директорию пользователя flenov. Ее адресом будет \\192.168.1.1\flenov.

Но здесь надо сделать одно замечание — в Linux обратная косая черта (`\`) является служебным символом, поэтому каждый такой знак должен заменяться двумя символами `\\`. Так как в начале UNC-имени идет `\\`, то они заменяются на четыре обратных косых черты, и адрес, приведенный выше, должен вводиться 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, необходимо указать имя пользователя, обладающего правами:

```
smbclient \\\\192.168.1.1\\flenov –U flenov
```

При удачном подключении к открытому ресурсу вы увидите такое приглашение:

```
Smb: \>
```

Теперь можно выполнять различные операции над файлами. Чтобы узнать, какие команды доступны, введите директиву `help` или знак вопроса (`?`). Команды, которые вы увидите, очень похожи на FTP (более подробно с FTP мы познакомимся в *главе 10*, см. также *Приложение 1*). Для отключения от ресурса необходимо выполнить команду `exit`.

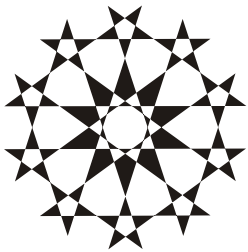
Большинство дистрибутивов Linux включают стандартный пакет Samba и ничего больше. А ведь в Интернете есть сторонние разработки, которые позволяют монтировать открытые в Samba ресурсы в файловую систему Linux так же, как дискету или CD-ROM, или работать с общими ресурсами в графическом режиме, как это делается в Windows. Однако имейте в виду, что это не всегда безопасно.

6.5. Развитие Samba

Хотя Samba имеет давнюю историю и вполне устойчивый код, он продолжает развиваться. Во время написания этой книги последняя стабильная версия программы имеет номер 3.5. Все версии начиная с 3.0 имеют возможность

подключаться к домену Active Directory, то есть к службе каталогов, продвигаемой и развиваемой фирмой Microsoft со времени выпуска Windows 2000 Server. Однако в качестве контроллера домена Active Directory сервер Samba пока выступать не может. В настоящее время разрабатывается версия 4.0 сервера Samba, которая будет лишена этого недостатка, и тогда можно будет полностью доверить Linux серверную часть сети, клиентские компьютеры которой работают под Windows.

Текущее состояние разработки Samba можно узнать на сайте проекта **www.samba.org**.



Глава 7

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- или HTTP-сервер. Уже долгое время самым распространенным из них является Apache. Трудно точно сказать, какая доля серверов работает на Apache, но можно с уверенностью утверждать, что их больше половины. Хотя для Linux существуют и другие WEB-серверы (например, TUX), когда говорят о WEB-сервере под Linux, подразумевают именно Apache. Да и в Windows можно очень часто встретить именно Apache, хотя в последнее время большую популярность стал получать Microsoft IIS.

Apache — абсолютно бесплатный сервер, который распространяется по лицензии GNU и доступен не только для UNIX-систем, но и для ОС Windows. Официальный сайт разработчиков — **www.apache.org**, где, помимо HTTP-сервера, можно найти еще ряд других проектов. Почему этот сервер стал так популярен? Неужели так повлиял фактор халявы? Бесплатность, конечно, соблазнительна, но качество превыше всего. Сервер Apache располагает громадным количеством возможностей и при этом обладает следующими немаловажными особенностями:

- безопасность — многие профессионалы относят его к разряду самых защищенных;
- надежность — в сочетании с ОС Linux WEB-сервер может работать без перезагрузок годами, а если и случаются перезагрузки, то, чаще всего, для обновления системы;

- неприхотливость — работает с минимальной нагрузкой на систему (не требует особых ресурсов);
- производительность — высокая скорость отклика и обработки запросов пользователей.

На основе Apache строят серверы, которые доступны 99,9% времени в год. Многие корпорации вверяют ему свои важные данные, и мне неизвестны случаи, когда кто-нибудь пожалел об этом. Да что там говорить, мои WEB-сайты работают у хостера на серверах с Apache.

Единственный недостаток, который отмечают пользователи, — это сложность конфигурирования. Приходится редактировать текстовый файл, а так как настроечных параметров огромное количество, то это может оказаться утомительным занятием. При этом очень сложно гарантировать, что все настройки будут сделаны оптимальными и безопасными, ведь очень легко упустить что-либо или банально опечататься. Да, в некоторых случаях конфигурационные файлы не так наглядны, как графические утилиты, но это дело привычки. Очень многие администраторы, наоборот, предпочитают именно текстовый формат настроек.

В этой книге мы не сможем рассмотреть все возможные параметры, потому что их слишком много. Мы коснемся только основных, и поговорим о том, что может сказаться на производительности и безопасности сервера. Рассматриваемого материала хватит в 99,9% случаев, и лично мне никогда не приходилось изменять ничего из того, что мы опустим.

7.1. Основные настройки

Основные настройки сервера Apache располагаются в файле /etc/httpd/conf/httpd.conf (для некоторых дистрибутивов — в /etc/httpd.conf). 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.

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 некоторые параметры игнорируются, и используются значения из конфигурации демона `inetd` (*см. разд. 5.4*);
- `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` — характер информации об установленном программном обеспечении, возвращаемый сервером. По умолчанию при любом обращении к серверу он возвращает заголовок с подробной информацией о системе, которая включает версии Apache, ОС Linux и всех имеющихся модулей. Если хакер узнает, что на сервере установлена старая версия интерпретатора PHP (или любой другой программы), то на взлом уйдет намного меньше времени. Болтливые параметры необходимо отключать, а информацию о сервере прятать. `ServerTokens` может принимать одно из следующих значений:
  - `Full` — отображать полную информацию о сервере и установленных модулях, включая их версии. Самое опасное для сервиса — это использование именно этого параметра;
  - `Min` — показать минимум сведений (только название сервера и установленные модули). Иногда даже такой простой список без версий может сказать хакеру слишком много;
  - `ProductOnly` — только название сервера, в нашем случае Apache вернет свое имя (apache) без указания версий. Вот это как раз то, что нам нужно.

    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 "2.2"
    ```

    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 устанавливаем только последние версии;

- `HostnameLookups` — флаг, отвечающий за проведение преобразования IP-адресов в доменные имена для логов и CGI-программ. Если установлено значение `on`, то IP-адрес клиента, запросившего данные с сервера, будет преобразован в доменное имя, иначе будет использоваться IP-адрес. Если в этом нет необходимости, для повышения быстродействия можно установить значение `off`;
- `User` и `Group` — имя пользователя и группа, с правами которого будет работать сервис. По умолчанию в Linux используется имя и группа 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` — местоположение журналов;
- `LogLevel` — степень подробности составления журнала. Можно указать одно из следующих значений: `emerg`, `alert`, `crit`, `error`, `warn`, `notice`, `info`, `debug`;
- `KeepAlive` — разрешение обрабатывать несколько запросов за одно соединение. По умолчанию этот параметр выключен (`off`), и для получения каждого файла требуется отдельное соединение. Это неэффективно и приводит к лишнему расходу ресурсов. Допустим, что пользователь запросил страницу с 10 картинками. В ответ на это браузер клиента откроет 11 соединений с WEB-сервером (одно для получения документа html и 10 — 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 очередного запроса от одного и того же клиента (в секундах). 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) допускает утечку памяти или других ресурсов, то во избежание проблем при длительной непрерывной работе сервера дочерний процесс при переполнении будет принудительно завершен. На большинстве систем это не требуется, но некоторые ОС (например, Solaris) страдают заметными утечками в библиотеках.

7.2. Модули

При настройке сервиса Apache очень важным звеном являются модули. Их загрузка 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 сценариев на языке Perl.

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. Я бы посоветовал загружать только то, что вам действительно нужно. 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 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 суперзащищенную программу, а на самом проверенном — наделать дыр. Это 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 важное правило: лишняя программа — враг администратора и дополнительная дверь для хакера.

Обязательно уделите внимание рассмотрению загружаемых модулей и удалите или закомментируйте то, что не нужно. 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 этом 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, такой:

```
<Location /server-status>
   SetHandler server-status
   Order deny,allow
   Deny from all
   Allow from .your-domain.com
</Location>
```

Первое объявление разрешает доступ 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**), разрешая его только клиентам из домена **your-domain.com**.

Если вы знакомы с HTML (HyperText Markup Language, язык разметки гипертекста) или XML (Extensible Markup Language, расширенный язык разметки), то такое объявление будет вам знакомо и более-менее понятно без

дополнительных пояснений. 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.

При задании прав доступа следует иметь в виду, что Apache просматривает разрешения и запреты для всех директорий, находящихся в иерархии выше запрошенной. При определении прав доступа к директории сперва определяются права доступа к корневой директории, которые затем могут быть переопределены правами доступа к любой поддиректории, лежащей в пути к запрошенному документу. Это позволяет эффективно управлять правами.

Поскольку мы думаем о безопасности, надо вспомнить наше основное правило — запрещено все, что не разрешено явно. Поэтому имеет смысл поместить в файл httpd.conf следующие строки:

```
<Directory />
   Order deny,allow
   Deny from all
</Directory>
```

которые запрещают доступ к любым файлам, а затем открыть доступ только к тем папкам, к которым он необходим. Теперь, если на сервере создать новую директорию, а права доступа к ней не прописать, то доступ будет закрыт, кроме тех случаев, когда директория создана внутри папки с уже открытым доступом.

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` — разрешает доступ к директории 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` — разрешает доступ, 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 использоваться два варианта:
  - `Order deny,allow` — изначально все разрешено, потом проверяются запреты, но они могут быть обойдены при помощи разрешений. 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 проверяются разрешения, он они могут быть обойдены запретами. Рекомендуется применять для всех директорий со сценариями или для всех каталогов, где находятся файлы, используемые определенным кругом лиц, например, администраторские сценарии;
- `Require параметр` — позволяет задать пользователей, которым разрешен доступ к каталогу. В качестве параметра можно указывать:
  - `user` — имена пользователей (или их ID), которым разрешен доступ к каталогу. Например, `Require user robert FlenovM`;
  - `group` — названия групп, пользователям которых позволен доступ к каталогу. Директива работает так же, как и `user`;
  - `valid-user` — доступ к директории разрешен любому аутентифицированному пользователю;
- `Satisfy параметр` — если указать значение `any`, то для ограничения доступа используется или логин/пароль, или IP-адрес. Для идентификации пользователя по двум условиям одновременно надо задать `all`;

- `AllowOverride параметр` — определяет, какие директивы из файла .htaccess в указанном каталоге могут перекрывать конфигурацию сервера. В качестве `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 параметры` — 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.

  В качестве аргумента `параметры` указывается `All`, что означает разрешение всех опций, кроме `MultiViews`, `None`, что запрещает все опции, или комбинация из следующих значений:

  - `ExecCGI` — разрешено выполнение CGI-сценариев. Чаще всего для этого используется отдельная директория /cgi-bin, но и в ней можно определить отдельные поддиректории, 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*, что ссылки сами по себе опасны, поэтому с ними нужно обращаться аккуратно, где бы они ни находились;
  - `SymLinksIfOwnerMatch` — следовать по символьным ссылкам, только если владельцы целевого файла и ссылки совпадают. При использовании символьных ссылок в 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 Includes, включения на стороне сервера);
  - `IncludesNOEXEC` — использовать SSI, кроме команд `exec` и `include`. Если вы не используете в сценариях CGI эти команды, то данная опция предпочтительнее предыдущей;
  - `Indexes` — отобразить список содержимого каталога, если отсутствует файл по умолчанию. Чаще всего пользователи набирают адреса в укороченном формате, например, **www.cydsoft.com**. Здесь не указан файл,

который нужно загрузить. Полный URL выглядит как **www.cydsoft.com/index.htm**, но если указан только домен, 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 ни один из таких файлов по указанному пути не найден, то при включенной опции `Indexes` будет выведен список содержимого каталога, а иначе — страница ошибки. Я рекомендую отключать эту опцию, потому что злоумышленник может получить слишком много информации о структуре каталога и его содержимом;

- `MultiViews` — представление зависит от предпочтений клиента.

Перед каждой из этих опций можно ставить знаки плюс или минус, что соответствует разрешению или запрещению опции. Если знак не указан, то это соответствует разрешению.

Все вышеописанные 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 определяться 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 совпадают с директивами для директорий. Поэтому 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 (что, разумеется, не имеет особого смысла и приведено только для примера, так как в таком случае разумнее просто удалить этот файл, разве что запрет носит временный характер).

Помимо файлов и директорий можно ограничивать и методы протокола HTTP, такие как `GET`, `POST`, `PUT`, `DELETE`, `CONNECT`, `OPTIONS`, `TRACE`, `PATCH`,

PROPFIND, PROPPATCH, MKCOL, COPY, MOVE, LOCK, UNLOCK. Где тут собака зарыта? Допустим, что у вас есть сценарий, которому пользователь должен передать параметры. Это делается одним из методов POST или GET. Если вы заведомо знаете, что программист использует только метод GET, то запретите все остальные, чтобы хакер не смог воспользоваться потенциальной уязвимостью в сценарии, изменив метод.

Бывают ситуации, когда не всем пользователям 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, это похоже на описание разрешений на файлы, даже права доступа используются те же самые. Они размещаются внутри определения директорий (<Directory> или <Location>) и влияют только на указанный каталог.

К примеру, так можно запретить любую передачу данных из директории /home в любом направлении:

```
<Directory /home>
   <Limit GET POST>
      Deny from all
   </Limit>
</Directory>
```

При этом внутри определения прав для директории /home устанавливается запрет на методы GET и POST.

Ваша задача как администратора — настроить параметры доступа на директории и файлы, при которых они будут минимально достаточными. Пользователь 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 каждого из которых используются на 1%.

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 она выключена (`off`), а пользователь не указал имя, то его аутентификация осуществляется другими модулями, например по IP-адресу;
- `AuthDBMGroupFile` и `AuthDBMUserFile` — аналогичны `AuthGroupFile` и `AuthUserFile`, но в качестве параметра указывается файл в формате базы данных Berkley-DB.

Эти команды помогут вам настроить идентификацию пользователей при обращении к определенным директориям. Например, если у вас есть каталог, работа с которым разрешена только авторизованным пользователям, то можно указать файл паролей и директиву `Require valid-user`, и при запросе файлов из этого каталога сервер предложит авторизоваться, указав имя пользователя и пароль.

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. 7.1 приведен пример такого окна.

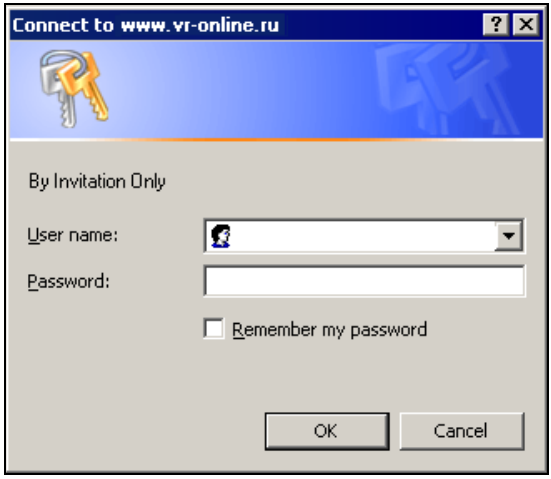


Рис. 7.1. Окно запроса имени и пароля

Директива `AuthUserFile` задает файл, в котором находится база имен и паролей пользователей сайта. Этот файл можно (и нужно) размещать вне области, доступной через WEB-сервер. 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 ранее (*см. разд. 7.3*).

Например, если нужно разрешить доступ только с определенного IP-адреса, то в файле может содержаться следующая строка:

```
Allow from 101.12.41.148
```

Если объединить защиту директивой `Allow from` и требование ввести пароль, то задача 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 /var/www/flenov/secret>
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, так как 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}1ZZEBtPy4/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 в том, что она может шифровать пароли и по алгоритму MD5, и с помощью системной функции `crypt()`. В одном файле могут находиться записи с обоими типами паролей.

Если вы храните имена и пароли в формате базы данных DBM (для ее указания в файле .htaccess используется директива `AuthDBMUserFile`), то для управления ею 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 рассмотрим ее основные ключи:

- `-c` — создать новый файл. Если указанный файл уже существует, то он перезаписывается, и старое содержимое теряется. Например, можно написать:

  ```
  htpasswd -c mypasswd robert
  ```

  После выполнения этой команды перед вами появится приглашение ввести пароль для нового пользователя robert и подтвердить его. В результате будет создан файл mypasswd, 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` — применить системную функцию `crypt()` для шифрования;
- `-s` — применить SHA-шифрование (на базе алгоритма хеширования), которое используется на платформе Netscape;
- `-p` — не шифровать пароли. Я не рекомендую использовать этот 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
```

Надо отметить, что хотя пароль можно задавать в командной строке, делать это не следует, так как такая команда небезопасна. Например, команда сохранится в истории командной оболочки, и незашифрованный пароль может таким (или другим) образом стать известным хакеру. Если пароль не задан, но требуется, то программа запросит его в интерактивном режиме.

У команды `htpasswd` есть два ограничения: в имени пользователя не должно быть символа двоеточия, и пароль не может превышать 255 символов. Оба условия достаточно демократичны, и с ними можно смириться. Из моих знакомых такой длинный пароль пока еще никто не захотел устанавливать, а использовать имя пользователя с двоеточием редко кому приходит на ум.

7.5.3. Проблемы авторизации

Авторизация — это слишком простой способ обеспечения безопасности. При передаче пароли шифруются простым кодированием Base64. Если хакер сможет перехватить пакет, содержащий имя пользователя и пароль, то он прочитает эту информацию через пять секунд. Для расшифровки Base64 не нужно подбирать пароль, достаточно выполнить одну функцию, которая 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. 7.9.*

7.5.4. Обработка на сервере

HTML-файлы могут обрабатываться прямо на сервере (так же, как выполняются файлы PHP). С одной стороны, это удобно, потому что код PHP можно будет вставлять в файлы с расширением htm, с другой стороны, HTML-файлы далеко небезопасны, и если хакер сможет их изменять, то сервер окажется под угрозой.

Чтобы разрешить серверу выполнять файлы с определенными расширениями на сервере, 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 нет необходимости использовать интерпретатор языка Perl, то первую строку следует закомментировать, чтобы она даже не смущала. Вторая строка безобидна, но если таким же образом разрешить серверу работать с HTM- или HTML-файлами, то это уже станет небезопасным. Следующей строки в вашем конфигурационном файле быть не должно:

```
AddHandler server-parsed .html
```

Если где-то действительно есть необходимость выполнять включения в HTML-документах, то пропишите это в файле .htaccess. В остальных 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 активна):

```
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 становится очень сложно. А чем больше параметров, тем выше вероятность, что вы что-либо прозеваете. Чтобы упростить поддержку вашего WEB-сервера, могу посоветовать следовать следующим рекомендациям:

- для удобства администрирования все описания прав доступа можно перенести в конфигурационный файл /etc/httpd/conf/access.conf. По умолчанию этот файл пуст, а все права доступа прописаны в /etc/httpd/conf/httpd.conf. Выделение части разрешений в отдельный файл действительно может помочь в управлении, потому что права в отдельном файле будут видны как на ладони;
- основные настройки сервера, которые редко изменяются, можно перенести в конфигурационный файл /etc/httpd/conf/basic.conf — такой файл надо создать;
- комментируйте все ваши действия. Многие настройки не изменяются годами, и уже через полгода трудно вспомнить, зачем вы установили определенную директиву. Например, вы запретили доступ для всех пользователей к какой-либо директории, которая временно использовалась для тестирования сценариев. Через какое-то время вы можете забыть об этом и случайно открыть доступ к сценариям, которые не были полностью отлажены и могут стать причиной взлома или крушения системы.

Если вы последовали первым двум рекомендациям, то нужно включить эти файлы в основной файл при помощи директивы `Include`, например

`Include /etc/httpd/conf/access.conf`. При этом стоит проверить, что такая директива не включена в файл по умолчанию.

Чем удобнее управлять безопасностью сервера, тем меньше ошибок вы совершите. Если все параметры хорошо видны, а подробные комментарии напоминают вам о назначении сделанных настроек, управление будет удобнее. Такой подход к администрированию также поможет 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 и любой ее поддиректории будут иметь указанные права, если они явно не переопределены для отдельных папок. Это значит, что под шаблон попадут такие папки, как /home/robert/public_html, /home/other_user/public_html и т. д.

7.7. Безопасность сценариев

Как мы уже говорили, сценарии являются очень опасными для WEB-сервера. Через их ошибки происходило очень много громких взломов. Мы уже знаем, что необходимо отключить все интерпретаторы, которые не используются, и оставить только то, что нужно. Этим мы усложним задачу хакера, но не решим проблему полностью.

Наиболее безопасный сайт — это тот, что использует статичные (HTML) документы и не имеет сценариев, выполняемых на сервере (PHP, ASP, Perl, Python и др.). Если вам нужен какой-то интерпретатор, то необходимо максимально ограничить его возможности.

Допустим, что ваш сайт использует сценарии PHP, и в этих сценариях есть функции, которые обращаются к системе. 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 не стоит надеяться на их профессионализм, потому что все мы — люди, и нам свойственно ошибаться. Администратор должен 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 выключаю безопасный режим лишь если это нереально.

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 умеет писать код, но чаще всего 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 про 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 возможностей. А что именно нужно сайту? Конечно же, это гостевая книга, форум, чат, голосование и т. д. Все эти разделы нельзя сделать с помощью языка разметки HTML, и нужны знания программирования, например, на Perl или PHP. Пользователи не хотят (или не могут) вникать в тонкости программирования, поэтому используют в своих проектах уже готовые (платные или бесплатные) движки.

Как я уже неоднократно говорил, любая разработка содержит ошибки, просто о них до поры до времени никому не известно. А распространенная программа становится лакомым кусочком для любого хакера, потому что ее взлом позволяет приникнуть сразу во множество систем, где она установлена.

Если администратор сайта устанавливает на свою страничку популярный форум, то он должен понимать, что 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 вы знаете 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 и начинающий хакер найдет уязвимость, даже 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, написанные 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 *главе 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 были 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 серьезности ошибки 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 отправляю для этого письмо анонимно.

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 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, и 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 страничек, очень часто использующие бесплатные или некачественные модули, в которых бывают ошибки, пользователи которых, к тому же, 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 **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;
- `SecFilterForceByteRange 32 126` — использовать символы только из указанного диапазона. Существует достаточное количество служебных символов (например, перевод каретки, конец строки), коды которых менее 32. Большинство из них невидимы, но требуют обработки нажатия соответствующих клавиш. Но как же тогда хакер может ввести такой символ в URL? Только через их код. Например, чтобы поместить в URL символ "перевод строки", необходимо указать в адресе "%13". В примере коды символов менее 32 и более 126 объявляются недопустимыми для адреса, и такие пакеты не пропускаются 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/passwd` — устанавливает запрет на использование в запросе пользователя подстроки "/etc/passwd". Таким же образом нужно добавить ограничение на файл /etc/shadow;
- `SecFilter /bin/ls` — отказ пользователям в обращении к директивам. В данном случае запрещается команда `ls`, которая может позволить хакеру увидеть содержимое каталогов, если в сценарии есть ошибка. Необходимо предотвратить обращения к таким командам, как `cat`, `rm`, `cp`, `ftp` и др.;

- `SecFilter "\.\./"` — классическая атака, когда в URL указываются две точки подряд для перехода в родительскую директорию. Их там быть не должно;
- `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).

Это основные методы, использование которых может повысить безопасность вашего WEB-сервера. Таким образом можно защищать целые 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 краткие рекомендации, которые помогут вам 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 корень системы, 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. Например, если на сервере есть PHP-скрипты, то пользователи видят только результат их выполнения. Чтобы посмотреть исходный код, хакеру необходим доступ к серверу, например FTP или Telnet, так как 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 такой файл, то он увидит его исходный код, что поможет быстрее найти ошибки в скриптах.

Когда злоумышленник исследует сценарии на сервере, то никто не мешает ему проверить наличие резервной копии. Увидев, что у вас есть файл **www.servername.com/index.php**, хакер попробует загрузить **www.servername.com/index.bak** или **www.servername.com/index.old**. На непрофессиональных сайтах такие версии встречаются очень часто. Не допускайте подобных промахов.

Любой специалист по безопасности должен запретить работу с резервными копиями. Сколько бы мы 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 безопасным, то есть запретить к ним доступ со стороны WEB-клиентов.

Чаще всего для резервных копий файлов используются те же имена файлов, но расширение меняется на .bak или .old. Следующий пример запрещает пользователям доступ к таким файлам:

```
<FilesMatch "\.bak$">
 Order deny, allow
 Deny from all
</FilesMatch>

<FilesMatch "\.old$">
 Order deny, allow
 Deny from all
</FilesMatch>
```

7.8. Индексация WEB-страниц

За последние 10 лет Интернет разросся до таких размеров, что найти в нем что-либо без хорошей поисковой системы стало невозможным. Первые такие системы просто индексировали страницы по их содержимому и потом использовали полученную базу данных для поиска, что давало очень приблизительные результаты. Если ввести в качестве контекста слово "лук", то будет

отобрано огромное количество сайтов по пищевой промышленности и по стрельбе из лука. В большинстве языков есть слова, которые имеют несколько значений, и поиск по ним затруднителен.

Проблема не только в двусмысленности некоторых слов. Есть множество широко употребляемых выражений, по которым тоже сложно произвести точную выборку. В связи с этим поисковые системы стали развиваться, и теперь можно добавлять в запрос различные параметры. Одной из самых мощных является поисковая система Google (**www.google.com**). В ней реализовано много возможностей, позволяющих сделать поиск более точным. Жаль, что большинство пользователей не освоили их, а вот взломщики изучили все 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 часто случается, что Google индексирует запрещенные страницы, и их можно просмотреть через поиск. Для этого всего лишь надо четко знать, какая информация хранится в файле, и как можно точнее составить строку поиска.

С помощью Google можно найти достаточно важные данные, которые скрыты от пользователя, но по ошибке администратора стали доступными для индексирующей машины Google. Во время поиска нужно правильно задавать параметры. Например, можно ввести в строку поиска следующую строку:

```
Годовой отчет filetype:doc
```

Или

```
Годовой отчет filetype:xls
```

И вы найдете все документы в формате Word и Excel, содержащие слова "Годовой отчет". Возможно, документов будет 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 прост и состоит всего лишь из двух директив:

- `User-Agent: параметр` — в качестве параметра передается имя поисковой системы, к которой относятся запреты. Таких записей в файле может быть несколько, и каждая будет описывать свою поисковую систему. Если запреты должны действовать на все поисковики, то достаточно указать 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 раскрыть секрет. 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 число зашедших по любым другим ссылкам или напрямую.

7.9. Безопасность подключения

Различные технологии прослушивания сетевого трафика в основном эффективны в локальных сетях, но хакеры больше любят интернет-соединения, потому что здесь можно найти больше интересного и есть лазейка, чтобы удаленно проводить атаку.

Что опасного может увидеть хакер, когда клиент просматривает страницы на WEB-сервере? Мы каждый день вводим на WEB-страницах какие-либо данные, пароли, номера кредитных карт, и именно это является основной целью хакера.

Как можно, например, находясь в Европе, перехватить трафик, который проходит между двумя городами в США? 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. 7.2). Таким образом, весь трафик будет проходить через компьютер хакера.

Рис. 7.2. Перехват трафика

Но этот метод был хорош несколько лет назад, когда не было HTTPS-протокола и безопасного соединения с помощью SSL-шифрования. Давайте

вспомним, что для подключения по SSL программа-клиент и сервер обмениваются ключами, с помощью которых происходит шифрование. Для HTTPS помимо открытого и закрытого ключей необходимы подтвержденные сертификаты, которые выдаются специализированными компаниями и подтверждают подлинность ключа. Программа-клиент проверяет сертификат, и если он достоверен (цифровая подпись принадлежит авторизованной фирме), то подключение разрешается. Сертификаты можно сгенерировать самостоятельно, а вот подпись подделать практически невозможно.

Если компьютер хакера 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.

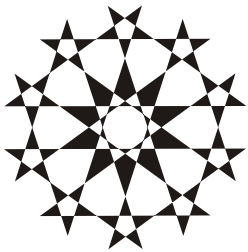
При такой схеме клиент получает ключ, который сгенерирован хакером и не имеет нужной подписи. Это значит, что у клиента появится сообщение о подключении к сайту без необходимого сертификата. И вот тут происходит самое страшное — большинство пользователей, которые давно работают с Интернетом, устали смотреть 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-серверы, в происхождении которых вы не уверены, ведь они могут 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 сообщение о подключении к сайту с неподписанным сертификатом сделать важным, то пользователи испугаются и прервут соединение.

С другой стороны, таких сайтов достаточно много, и многие из них вполне уважаемы и защищены. Просто за подпись надо платить, а не каждый захочет платить за проекты, которые не будут приносить денег. Таким образом, подписанными чаще всего являются коммерческие проекты. Но даже у платных и авторитетных ресурсов бывают проблемы. Сертификат действителен только в течение указанного в нем периода, и если администратор не уследил за датой, то он устареет.

Единственное, что должен помнить пользователь: при неавторизованном соединении нельзя вводить действительно секретной информации, например, 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 или напарника в игре, а многие используют сеть для работы или обучения. Но все мы не можем жить без общения, и, несмотря на новые технологии, которые выдумывают для облегчения общения (чаты, IRC, ICQ и т. д.), электронная почта жила, живет и будет жить. Именно с электронной почты начали развиваться сети, и она была одним из первых сервисов Интернета.

Лично для меня почтовый клиент стал основной программой, которой я пользуюсь чаще всего. Я веду переписку с читателями, друзьями, начальством и т. д. Так уж получилось, что люди, с которыми я работаю, находятся в других городах и даже странах. Расстояние до самых ближайших партнеров более 1000 км, а до издательства несколько лет назад было около 1700 км (сейчас я уже переехал в Питер, и мой дом расположен в паре километров от издательства). Если раньше это было сопряжено с большими проблемами, то теперь, благодаря компьютеру и Интернету, я могу, к примеру, жить в теплых краях, а выполнять работы для компании, которая находится на Аляске. Таким образом, легко и сохранить ноги в тепле, и зарабатывать деньги в холодных областях. Я же почему-то пару лет назад, наоборот, переехал из теплых краев на север. Не на дальний, а в северную столицу из южной.

Как работает электронная почта (E-Mail)? Рассмотрим основные моменты доставки письма:

1. Пользователь создает в почтовом клиенте (программе электронной почты) письмо, указывает получателя и отправляет его почтовому серверу. В настоящее время для передачи сообщений чаще всего используются SMTP-серверы, а для работы с ними — протокол SMTP.
2. Сервер, получив письмо, определяет место назначения, то есть имя сервера. Адрес состоит из двух частей: имени пользователя и имени сервера,

разделенных между собой знаком @, например, **djon@servername.com**. Здесь djon — это имя пользователя, а servername.com — имя сервера. С помощью DNS SMTP-сервер узнает IP-адрес сервера (в нашем случае servername.com), которому должно быть доставлено письмо.

3. Письмо направляется серверу, на котором зарегистрирован получатель.
4. Получив письмо, сервер servername.com проверяет наличие адресата (у нас это пользователь djon), и если он присутствует, помещает письмо в его почтовый ящик. Если указанного пользователя нет, сервер сообщает об этом отправителю, возвращая письмо или его часть.
5. Пользователь просматривает свой почтовый ящик с помощью почтового клиента и может скачать письмо для чтения.

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 почтовое отделение 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. Многие с удовольствием заменили бы этот протокол чем-нибудь более безопасным, но нынешние серверы слишком популярны, чтобы в один миг взять и отказаться от них.

Несколько десятков лет назад для работы с почтой широко использовался протокол UUCP (UNIX to UNIX Copy, копирование между UNIX-системами). Но он был слишком сильно привязан к ОС и при этом обладал ограниченными возможностями, поэтому не получил распространения и в настоящее время практически не используется.

Для приема почты используются три протокола:

- POP3 — Post Office Protocol v3 (Почтовый протокол), в настоящее время наиболее распространенный протокол приема почты;
- IMAP4 — существуют две интерпретации этого сокращения: Internet Message Access Protocol (Протокол доступа к сообщениям в сети Интернет) и Interactive Mail Access Protocol (Протокол интерактивного доступа к электронной почте). Этот протокол обладает бóльшими возможностями по сравнению с POP3, например, позволяет управлять сообщениями прямо на сервере, без загрузки текста на компьютер пользователя, а лишь загружая заголовки писем;
- MAPI — Messaging Application Programming Interface (Интерфейс прикладного программирования электронной почты), который используется в сетях Microsoft на серверах Microsoft Exchange.

Кроме того, сейчас очень распространены базирующиеся в WEB системы просмотра почты, когда пользователи вовсе не скачивают почту на свой компьютер, а просматривают ее, используя браузер.

Самым распространенным средством доставки почты в Linux является самая старая программа sendmail. Она обладает большими возможностями, но достаточно сложна в использовании. Из-за почтенного возраста в сервере sendmail сохранилась и возможность передачи по протоколу UUCP, которая сейчас используется крайне редко.

Принцип работы sendmail достаточно прост. Получив письмо от клиента, программа определяет получателя и заносит необходимую для доставки служебную информацию в заголовок письма. Дальнейшие действия зависят от настроек. Например, письмо может быть отослано немедленно или помещено в хранилище. Через определенные промежутки времени накопившиеся письма отправляются своим адресатам.

8.1. Настройка *sendmail*

Основной конфигурационный файл, который вам понадобится — /etc/sendmail.cf. Сервер sendmail имеет плохую репутацию в связи со сложностью настройки. Действительно, если лишь посмотреть на размер файла /etc/sendmail.cf, то становится жутко: это 32 Кбайт (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, в частности, обычно присутствуют следующие:

- `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 цель — 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 этого 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
dnl file, you need the sendmail-cf rpm installed and then have to
dnl generate a new /etc/sendmail.cf by running the following command:
dnl Это файл макроконфигурации sendmail. Если вы делаете изменения в этом
dnl файле, вам нужна инсталляция rpm sendmail-cf, с помощью которой можно
dnl сгенерировать новый /etc/sendmail.cf, запустив команду:
dnl
dnl        m4 /etc/mail/sendmail.mc > /etc/sendmail.cf
dnl
include(`/usr/share/sendmail-cf/m4/cf.m4')
VERSIONID(`linux setup for ASPLinux')dnl
OSTYPE(`linux')
dnl Uncomment and edit the following line if your mail needs to be
dnl sent out through an external mail server:
dnl Расскомментируйте следующую строку, если вам нужно посылать почту
dnl через внешний почтовый сервер:
dnl define(`SMART_HOST',`smtp.your.provider')
define(`confDEF_USER_ID',``8:12'')dnl
undefine(`UUCP_RELAY')dnl
undefine(`BITNET_RELAY')dnl
...
...
```

Файл sendmail.mc имеет более простой формат по сравнению со старым sendmail.cf, благодаря чему уменьшается вероятность допустить ошибку при конфигурировании. После исправлений его необходимо скомпилировать (перевести файл sendmail.mc в формат sendmail.cf) специальной командой, в результате чего получается файл /etc/sendmail.cf.

Мы в основном будем рассматривать параметры, которые нужно устанавливать в файле sendmail.cf, а если описывается параметр файла sendmail.mc, я буду явно указывать на это.

Иногда при неправильных настройках загрузка Linux может зависать при старте сервиса sendmail. Это происходит из-за того, что ваш почтовый сервер не может определить имя компьютера. Откройте файл /etc/hosts. В нем чаще всего будет только одна строка:

```
127.0.0.1       localhost.localdomain       localhost
```

Подробно этот файл будет описан в *главе 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. Но на деле все сетевые обращения происходят не по имени, а по IP-адресу, и если sendmail не может определить адрес по имени FlenovM, происходит зависание. Чтобы исправить эту ситуацию, добавьте в файл /etc/hosts строку:

```
192.168.77.1        FlenovM        FlenovM
```

Только FlenovM нужно заменить на имя вашего компьютера и, разумеется, указать свой IP-адрес. После этого программу sendmail можно помещать в автозапуск, и она будет работать великолепно даже с настройками по умолчанию.

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 файле /var/spool/mail/root.

Для работы с почтой нам нужен почтовый клиент, который будет отправлять письма серверу и принимать от него новые сообщения. Таких программ существует множество, и в некоторых дистрибутивах Linux предлагается аж 7 различных клиентов. Какой выберете вы, зависит только от личных предпочтений.

В принципе, можно обойтись и без клиента, а соединение с сервером осуществлять напрямую через сервис Telnet, благо команды SMTP достаточно просты и их легко использовать. В этом случае для отправки почты нужно подключиться к порту 25 (порт SMTP), а для получения — к 110 (порт POP3).

8.2. Работа почты

Рассмотрим работу ящиков на примере почтового клиента KMail — графической программы, которая должна у вас присутствовать, если установлена KDE. Для его запуска из главного меню Linux выберите команду **Internet | KMail**. Перед вами откроется окно, представленное на рис. 8.1.

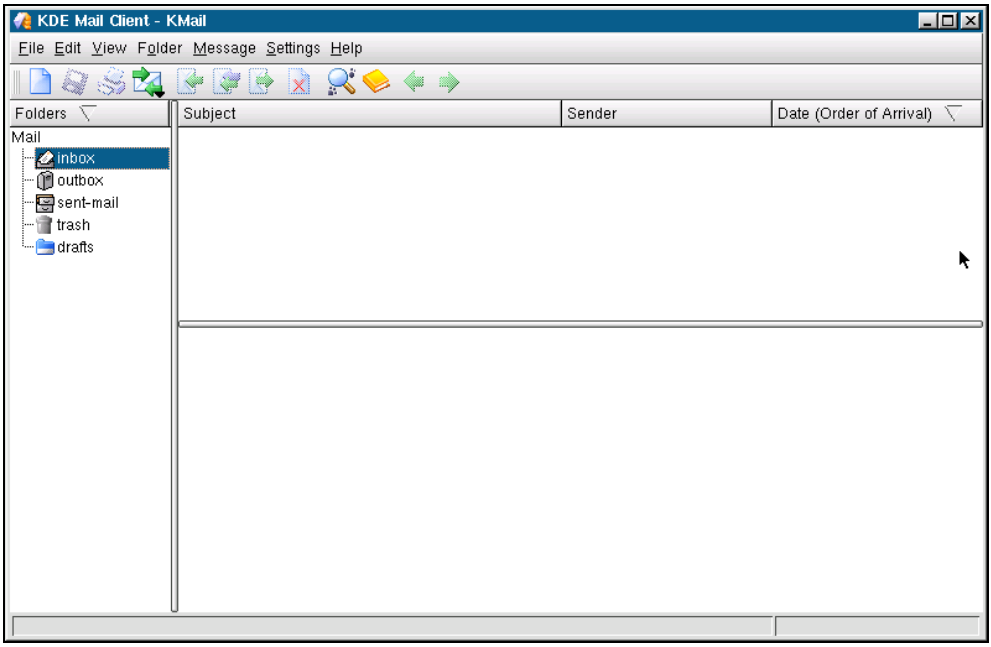


Рис. 8.1. Главное окно программы KMail

Программа еще не знает, с каким почтовым ящиком мы хотим работать, и это необходимо настроить. Воспользуйтесь пунктом меню **Settings | Configure KMail**. Перед вами откроется окно конфигурации. В нем выберите раздел **Network**, и вы увидите окно с двумя вкладками: **Sending** и **Receiving** (рис. 8.2).

На вкладке **Sending** мы должны указать параметры сервера, через который будет происходить отправка. По умолчанию уже настроен локальный sendmail, но что делать, если почтовый сервер расположен на другом компьютере? Давайте удалим существующую запись (выделите и нажмите кнопку **Remove**), а потом создадим свою.

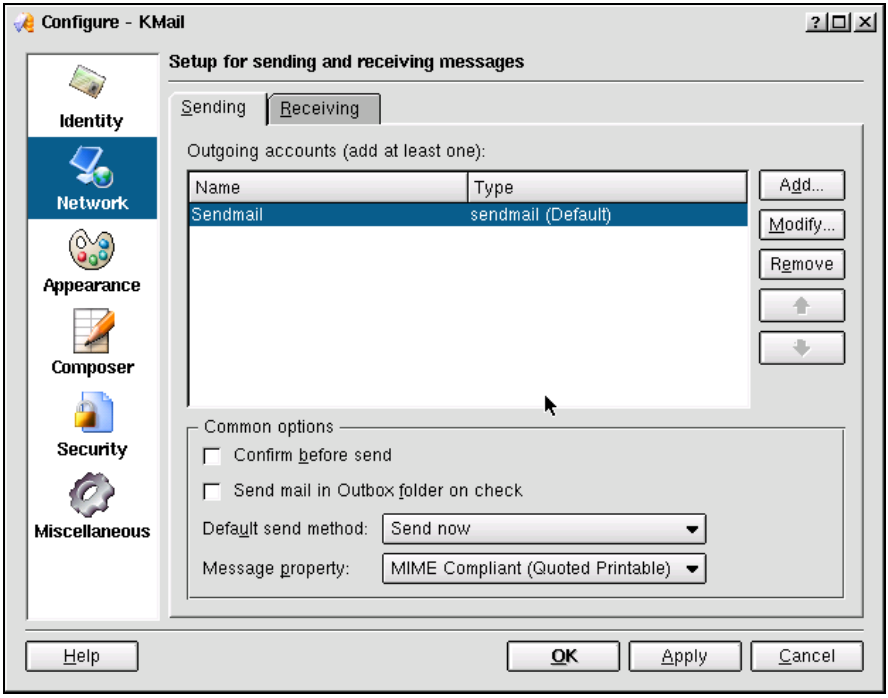


Рис. 8.2. Окно сетевых настроек

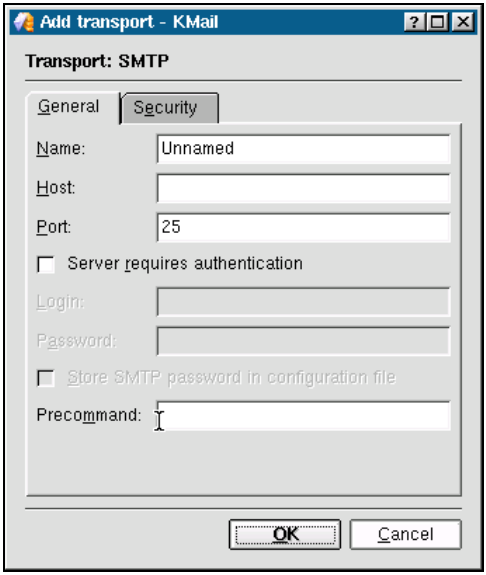


Рис. 8.3. Окно настроек SMTP-сервера

Нажмите кнопку **Add** для добавления новой записи. Перед вами появится окно выбора протокола: SMTP или Sendmail. Выбираем SMTP как наиболее универсальный вариант. Теперь откроется окно, в котором нужно указать параметры SMTP-сервера, через который мы будем посылать почту (рис. 8.3).

В этом окне 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 порт 25, 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 настройки параметров SMTP-сервера не должен вызвать проблем.

Теперь рассмотрим настройку сервера для чтения почты. Перейдите на закладку **Receiving**, и вы увидите список серверов. Выделите имеющиеся записи и удалите их. Затем 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 всего используется POP3, и процесс создания записи для работы с ним похож на аналогичный процесс для SMTP-сервера. 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 локальным 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 удобнее всего использовать почтовый клиент.

Итак, создадим новую учетную запись, чтобы можно было в удобном виде читать сообщения по безопасности. Нажмите кнопку **Add**, и перед вами появится окно выбора типа сервера. Выберите тип ящика **Local mailbox** и нажмите **OK**. Перед вами появится окно создания нового аккаунта, как на рис. 8.4.

Здесь необходимо заполнить следующие поля:

- **Name** — имя записи, которое может быть любым;
- **Location** — расположение локального ящика. По умолчанию все ящики хранятся в файле /var/spool/mail/имя, где имя — это имя пользователя. Для администратора root нужно указать размещение /var/spool/mail/root.

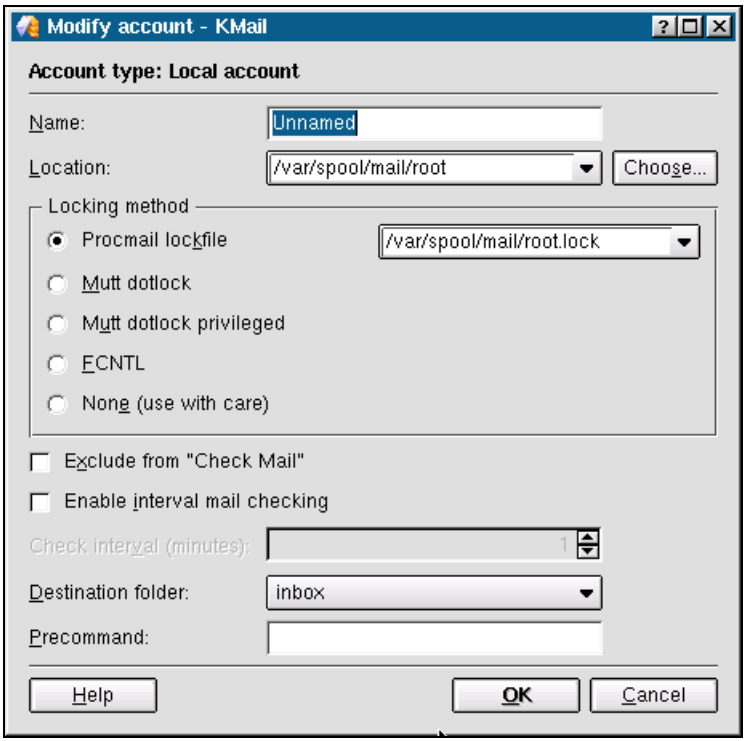


Рис. 8.4. Окно настроек аккаунта типа Local mailbox

Остальные параметры, чаще всего, остаются заданными по умолчанию, если администратор не натворил чего-то особого в конфигурации.

Попробуйте прочитать почту разными протоколами. Убедитесь, что все работает верно, сообщения приходят на ваш почтовый ящик и доходят до получателя. Если указаны настройки по умолчанию, все должно работать. В дальнейшем мы рассмотрим некоторые специфичные настройки, которые помогут сделать ваш почтовый сервер более безопасным, но прежде чем приступать к улучшениям, нужно убедиться, что работает базовый вариант.

8.2.1. Безопасность сообщений

Сообщения E-mail пересылаются по сети в виде простого 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 Internet) — этот стандарт шифрования поддерживается в основном почтовыми клиентами Netscape и его клонами. Это накладывает некоторые ограничения, ведь не все пользователи привыкли использовать именно эти программы;
- PGP (Pretty Good Privacy, "достаточно хорошая приватность") — программа шифрования, которая 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 GnuPG. Нет, она не лучше других, потому что все PGP-клоны используют один и тот же принцип. Просто GnuPG разработан за пределами США, где 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 и количества байт;
- `sendmail` — запустить сервер sendmail. Используя ее с различными ключами, можно увидеть достаточно много полезной информации. За более подробной справкой следует обратиться к документации `man`.

8.4. Безопасность sendmail

Безопасность sendmail далека от идеала, и в нем регулярно находят ошибки. По этому поводу администраторы и программисты начали слагать анекдоты и превратили сервис в объект для насмехательств. Я слышал, что некоторые даже делали ставки на то, будет ли 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

Проблемы безопасности начинаются еще на этапе подключения. Сервис sendmail, как и большинство других служб, выдает строку приветствия, в которой содержится информация об имени и версии программы.

Хакер не должен знать этих данных. 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 теперь значение этому параметру присваивается следующим образом:

```
SmtpGreetingMessage=$j $b
```

Если ваша система сообщает о себе 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 должен принимать писем, то необходимо запретить режим приема. Для этого откройте файл /etc/sysconfig/sendmail и измените его содержимое:

```
DEAMON=yes
QUEUE="q1h"
```

Вторая строка задает параметры, которые будут передаваться программе sendmail при запуске. Чтобы возобновить прием почты, измените значение на `bd`.

Если у вас нет файла /etc/sysconfig/sendmail (он используется не во всех дистрибутивах), то придется редактировать сценарий /etc/rc.d/init.d/sendmail. Найдите в этом файле параметры, которые передаются программе, и измените их на `q1h` прямо в тексте сценария.

8.4.3. Права доступа

Ни один сервис в ОС не должен работать от имени администратора root. Если в коде программы будет найдена лазейка, позволяющая запускать команды, то можно считать систему потерянной, потому что директивы будут выполняться с правами root. Опытные пользователи компьютеров, наверное, помнят, что несколько лет назад в sendmail чуть ли не каждую неделю находили ошибки, и большинство из них были критичными.

Сервис должен работать с правами пользователя, которому доступны только необходимые для работы 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 знаком `#` в начале строки. 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 для сервера может оказаться 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 адрес E-mail или даже имя пользователя. С помощью этой директивы хакеры могут собирать списки для рассылки спама. Не стоит давать в руки злоумышленника такую информацию.

8.4.5. Выполнение внешних команд

В почтовом сервисе есть одна серьезная проблема — ему необходимо выполнять системные команды, а это всегда опасно. Если хакер сможет запустить такую команду без ведома администратора и с повышенными правами, то это грозит большими неприятностями. Именно поэтому мы понижали права, с которыми работает сервис, но этого недостаточно.

Чтобы запретить выполнение системных команд, необходимо заставить 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 каталогов запускать невозможно. Если в каталоге /usr/adm/sm.bin находятся только безопасные программы, то ваша система наименее подвержена уязвимости.

8.4.6. Доверенные пользователи

В сервисе sendmail можно создать список пользователей, которым вы доверяете 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, подключения на порты 25 и 110, чаще всего, общедоступны.

Для защиты сервера от DoS-атак со стороны хакера нам помогут следующие параметры сервиса sendmail:

- `MaxDeamonChildren` — ограничение количества одновременно запущенных процессов. С помощью этого параметра мы можем защитить ресурсы сервера (процессор) от излишней перегрузки. По умолчанию установлено значение 12, но для мощного компьютера его можно повысить, чтобы эффективнее использовать процессор, а для слабого — уменьшить;
- `ConnectionRateThrottle` — максимальное количество открываемых соединений в секунду. По умолчанию этот параметр равен 3, и повышать его без особой надобности не стоит, разве что 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. Это классическая атака DoS, только в отношении почтового ящика.

На первый взгляд необходимо просто увеличить размер ящика или убрать лимит вовсе. Это неверное решение, поэтому забудьте про него. Если ящик не будет иметь предела, то хакер сможет произвести атаку DoS на весь сервер.

Почтовые сообщения — единственный способ закачать информацию на сервер. Когда злоумышленник отправляет E-mail, оно сохраняется на сервере, пока не будет скачано пользователем. Слишком большое количество информации займет все пространство на жестком диске, и сервер больше не сможет принимать сообщения ни на один из почтовых ящиков.

Самая худшая ситуация при почтовой бомбардировки 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.

Ограничение на используемое под хранение писем пространство необходимо. Лучше потерять контроль над одним почтовым ящиком, чем над всем почтовым сервером.

От почтовой бомбардировки защититься нельзя, но можно попытаться усложнить задачу злоумышленника. Для этого нам понадобятся параметры, которые мы рассматривали в *разд. 8.4.7*. Кроме того, желательно ограничить максимальный размер сообщения до приемлемых пределов с помощью параметра `MaxMessageSize`. Если это сделать, злоумышленнику понадобится направлять на сервер множество маленьких писем вместо нескольких больших. Однако, к сожалению, большие письма иногда отправляют не только хакеры, и этим вы усложните жизнь и реальным корреспондентам.

8.6. Спам

Проблема XXI информационного века — рассылка нежелательной корреспонденции. Это действительно болезнь, с которой надо бороться, а существующие методы пока не приносят необходимого результата, и спам отнимает бóльшую часть почтового трафика.

Один из способов борьбы с нежелательной корреспонденцией — запрет серверов, с которых приходит спам. Но те, кто занимаются такими рассылками, находят все новые пути для обхода барьеров, в том числе 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:

- излишние расходы на трафик, которые постоянно увеличиваются;
- отвлечение внимания 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, но 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('dnsbl', 'spam.com', ' 550 Mail not accepted from this do-
main')dnl
```

Версии 8.11 и более поздние:

```
HACK('check_dnsbl', 'spam.com', '', 'general', 'reason')dnl
```

В обеих строках `spam.com` нужно заменить на адрес сервера DNSBL — сервера, на котором хранятся списки IP-адресов, с которых рассылается спам, так называемые спам-листы. Список таких серверов можно посмотреть по адресу **http://spamlinks.net/filter-dnsbl-lists.htm**. При наличии адреса в списке соединение с ним блокируется. Этот метод не очень эффективен, потому что спам-листы, разумеется, содержат адреса лишь некоторых, уже сильно наследивших спамеров. К тому же, напротив, были случаи, когда ошибочно блокировались безобидные серверы.

Однажды мой сервер, с которого происходит продажа программного обеспечения, был заблокирован в одном из спам-листов. Это были времена, когда в черные списки попадали и по делу, и просто так. Когда я рассылал своим пользователям их регистрационные ключи, то 10% сообщений возвращалось. 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 определяется, что письмо содержит спам, то оно удаляется.

Этот способ более эффективен, но по тексту очень сложно определить, является ли письмо рекламной рассылкой. Хакеры постоянно ищут новые пути обхода таких блокировок, поэтому процент фильтрации невысок. Вы можете настроить программу так, что она будет уничтожать все сообщения, в которых есть слова "купи", "продаю" и тому подобные 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 (**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 100 получателями 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 кому угодно. Чтобы избежать этого, можно выполнить одно из следующих действий:
  - запретить с помощью сетевого экрана подключение к порту SMTP пользователей, которые находятся вне вашей сети. Такую защиту чаще всего используют провайдеры и администраторы частных или корпора-

тивных сетей. С реализацией этого запрета у вас не должно возникнуть проблем, потому что мы уже не раз ее рассматривали;

- разрешать отправку почты только в течение определенного времени (например, 10 минут) после проверки почты по протоколу POP3. В момент проверки почты сервер производит авторизацию клиента, и по этим данным 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 реализовать авторизацию и сделать ее обязательной;

❐ запретить отправку слишком большого количества писем с одного и того же IP-адреса. Вполне нормальным числом является 20. Пользователь не должен иметь права посылать более 20 писем за 10 минут;

❐ запретить отправку писем большому количеству получателей. Письмо может содержать в поле "CC" список пользователей, которым направлено письмо.

Существуют и другие методы, но даже этих будет достаточно.

В последних версиях sendmail по умолчанию разрешается пересылка почты только с тех компьютеров, которые прописаны в файле /etc/mail/access. В листинге 8.2 приведено содержимое этого файла.

Листинг 8.2. Файл /etc/mail/access

```
# Check the /usr/share/doc/sendmail/README.cf file for a description
# of the format of this file. (search for access_db in that file)
# The /usr/share/doc/sendmail/README.cf is part of the sendmail-doc
# package.
#
# Смотрите файл /usr/share/doc/sendmail/README.cf для получения
# информации по формату файла (ищите слово access_db в этом файле)
#
# by default we allow relaying from localhost...
# По умолчанию мы разрешаем рассылку только с локального хоста
localhost.localdomain              RELAY
localhost                          RELAY
127.0.0.1                          RELAY
```

В этом файле можно оставить разрешение отсылать почту только с компьютеров внутренней сети или с самого сервера. Для этого файл должен содержать следующие записи:

```
localhost               RELAY
your_domain.com         RELAY
```

Все остальные не смогут выполнять рассылку писем. Такой метод хорош только в том случае, когда серверы и компьютеры защищены. Если хакер проник в сеть, то он сможет разослать любой спам от имени пользователя сети. От этого никуда не деться. Если есть хоть какое-то разрешение, то им можно воспользоваться, и ваша задача сделать так, чтобы это было, по крайней мере, сложно.

Запрет рассылки не всегда может быть реализован, поэтому приходится использовать другие методы, например, заставить пользователей проверять почту по протоколу POP до отправки писем. Для осуществления этого метода можно воспользоваться сервисом pop-before-smtp (**popsmtp.sourceforge.net**), который проверяет журнал сообщений /var/log/maillog, и отправка почты разрешается, только если в нем найдена удачная авторизация за определенный период.

Единственный, но существенный недостаток этого способа заключается в том, что если на пути следования письма стоит анонимный прокси-сервер или маскирующий сетевой экран, то пакеты будут приходить с другим IP-адресом. Это значит, что все пользователи, которые подключены через тот же Proxy или Firewall, также автоматически считаются авторизованными. Таким образом, поиск по IP-адресу записей в журнале не может дать полной защиты.

Наиболее предпочтительным является метод, при котором используется SMTP-авторизация (`SMTP AUTH`), которая описана в RFC 2554. В сервисе sendmail поддержка этого расширения существует еще с версии 8.10.

Если вы решили использовать `SMTP AUTH`, то убедитесь, что почтовые клиенты пользователей имеют возможность авторизации на сервере и при этом настроены на ее использование.

8.7. Postfix

В моей любимой Mandriva используется более простой в настройке почтовый сервер, который называется Postfix. Его можно также встретить в Debian и Suse. За конфигурацию данного сервера отвечает файл /etc/postfix/main.cf. Здесь параметров не так много, потому что вся конфигурация сервера разбита на несколько файлов, и main.cf является главным, но не единственным. Например, более полная версия этого файла — это main.cf.dist.

Мы рассмотрим лишь базовые параметры. Знание английского и хорошие комментарии в файле конфигурации помогут вам разобраться со многими значениями, если вдруг понадобится что-то из того, что мы не рассмотрели. Я же постарался выбрать наиболее важные параметры, которые покроют максимальное количество ваших потребностей.

Первый параметр, который может пригодиться — `myorigin`. Как и большинство базовых параметров, этот можно найти в файле main.cf.dist. Там можно найти аж две строки следующего вида:

```
#myorigin = $myhostname
#myorigin = $mydomain
```

Обе строки закомментированы, они лишь предлагают вам выбрать тот вариант, который лучше подходит. Чтобы одну из строк сделать активной, просто убираем символ комментария # в начале строки. После символа равенства идет значение, но что же это за интересные значения, начинающиеся с символа доллара? Это переменные. Все, что начинается с доллара — это переменные, и вы можете создавать собственные переменные и назначать им значения. Например, в следующей строке кода я создал переменную `$siteofflenov` со значением `www.flenov.info`:

```
$siteofflenov = www.flenov.info
```

Теперь посмотрим на переменные, которые мы недавно увидели: `$myhostname` и `$mydomain`. По имени уже можно понять, что первая равна имени хоста, а вторая домену. Так что же означает параметр `myorigin`? Он определяет, как сервер будет представляться при работе с другими системами.

Следующий параметр — `inet_interfaces`. С его помощью можно указать интерфейсы, с которых нужно ожидать почту. По умолчанию сервер ожидает письма со всех активных интерфейсов, и параметр равен значению `all`. Но иногда необходимо запретить определенные интерфейсы.

Параметр `mydestination` определяет домены, для которых сервер принимает почту.

8.7.1. Псевдонимы

Для настройки псевдонимов используется файл aliases. В этом файле находятся записи вида

```
admin:  root
```

Слева от двоеточия стоит алиас, а справа можно увидеть имя пользователя. В данном случае имя admin является алиасом для root, или, другими словами, все письма, которые будут приходить на имя admin, будут автоматически пересылаться на ящик root.

Нужно иметь в виду, что aliases является конфигурационным и текстовым, а сервер использует бинарную версию aliases.db. Чтобы создать этот db-файл, необходимо выполнить команду

```
postalias aliases
```

Но и это еще не все. Изменения вступят в силу не сразу, а только по прошествии какого-то времени. Если вы торопитесь и хотите получить результат немедленно, можно перезапустить сервер. Это помогает всегда. Но неужели после каждого изменения нужно перезапускать программу? Конечно же, нет, это не самое лучшее решение, потому что есть способ проще — выполните следующую команду:

```
postfix reload
```

8.7.2. Ретрансляция

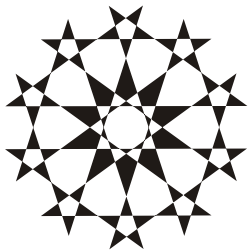
Сервер Postfix может передавать письма, только если:

- отправитель из одной из сетей, указанных в переменной $mynetworks. По умолчанию эта переменная равна всем сетям, которые подключены к данному компьютеру, например:

  ```
  mynetworks = 168.100.189.0/28, 127.0.0.0/8
  ```

- домен отправителя или домен получателя сообщения присутствует в переменной `$relay_domains`.

Такая защита необходима, чтобы спамеры не использовали ваш почтовый сервер в качестве средства распространения спама. За спам в наше время сильно наказывают. Нет, в тюрьму сажают редко, но в игнор помещают без разговора. А если сервер попадет в список игнорируемых, то вы не сможете отправлять никаких сообщений.



Глава 9

Шлюз в Интернет

Мы уже знаем, как установить и настроить сервер Linux, сделать его соединение безопасным и подключить основные сервисы. Но такие службы как WEB-сервер и электронная почта используются не только в локальной сети. Их максимальные преимущества проявляются при интеграции с всемирной сетью.

Но мир 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 окна и двери, думаем об их надежности, а закончив отделку дома, ставим сигнализацию. Дом мы уже построили и сделали его защищенным. Именно поэтому безопасность сети рассматривалась первой. Теперь нам необходимо поставить двери, через которые можно будет выходить из дома на улицу (в Интернет). После завершения всех настроек мы повесим на нашу обитель сигнализацию — системы мониторинга и выявления атак, которые будут рассматриваться в *главе 12*.

В качестве дверей во всемирную сеть будут выступать шлюз и прокси-сервер (Proxy). Именно о них пойдет речь в этой главе, и как раз их мы будем подробно рассматривать. Но настройка связи не заканчивается конфигурированием сервера, на клиенте тоже необходимо производить определенные действия, с которыми нам также предстоит познакомиться.

9.1. 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, значит, сервис преобразования имен доступен, иначе с Интернетом можно будет работать, только используя IP-адреса. Если имя не определяется, то это можно исправить, вручную прописав адрес DNS-сервера. Получите эту информацию у своего интернет-провайдера и добавьте следующую строку в файл /etc/resolv.conf:

```
nameserver 192.168.1.1
```

Адрес 192.168.1.1 нужно заменить тем, который вам предоставил провайдер. Если вам дали несколько адресов, то можно указать их все, каждый в своей строке:

```
nameserver 192.168.1.1
nameserver 192.168.1.2
```

9.2. Работа прокси-сервера

Изначально прокси-серверы (Proxy) создавались для решения узкого круга задач, а именно, для кэширования данных, получаемых из Интернета. Например, сотрудники вашей фирмы работают на 100 компьютерах, которые подключаются к Интернету через один физический канал связи. Не секрет, что бóльшая 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 подсчитайте, сколько происходит загрузок сайта **www.yahoo.com** со 100 компьютеров. Получится число более 1000, потому что в среднем человек обращается к поисковикам не менее 10 раз в день для поиска разной информации и уточнения запросов. При этом если сами страницы меняются в зависимости от запроса, то, скажем, картинки остаются одними и теми же. 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 этой главе мы поговорим о самом популярном в Linux прокси-сервере squid, рассмотрим его возможности, оценим безопасность и познакомимся с конфигурированием.

Чтобы сэкономить трафик и одновременно увеличить скорость загрузки на сервере, через который осуществляется выход в Интернет, устанавливается специализированная программа (прокси-сервер), которая кэширует весь трафик (рис. 9.1). Когда первый пользователь загружает страницу **www.yahoo.com**, то все ее содержимое сохраняется в кэше прокси-сервера. При следующем обращении к WEB-узлу все картинки скачиваются уже не из Интернета, а с прокси-сервера, а текстовая часть (в зависимости от содержимого страницы и происшедших изменений) может быть загружена с сервера.

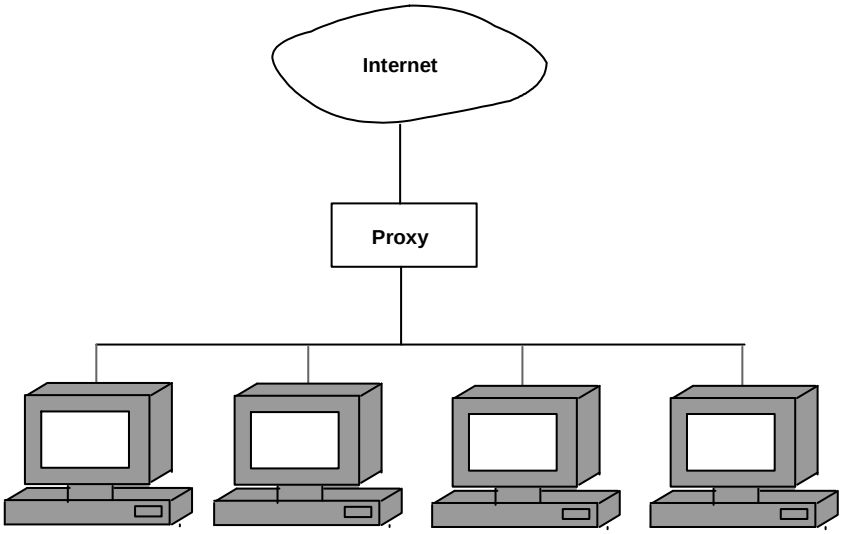


Рис. 9.1. Организация работы с Интернетом через прокси-сервер

Как правило, на сайтах именно графический материал занимает наибольший объем. Если текстовая часть страниц обычно не превышает 15 Кбайт, то общий размер графики достигает 100 Кбайт и более. Загружая эту информацию с локального прокси-сервера, вы экономите трафик и время.

Скорость загрузки увеличивается за счет того, что прокси-сервер находится в вашей локальной сети, и связь с ним, чаще всего, быстрая. Реальная скорость соответствует вашему оборудованию, пропускная способность которого в настоящее время даже в самых дешевых вариантах достигает 100 Мбит/с. По этому каналу вы забираете бóльшую часть информации (всю графику, флеш-анимацию и неизмененную текстовую часть). Связь с Интернетом намного медленнее, 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, прокси-сервер может сохранять результаты DNS-запросов. Это также может повысить производительность. Пользователю удобнее вводить символьные адреса, а компьютер обменивается данными, используя 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 задержка на работу с сервером DNS будет меньше, потому что прокси-сервер возьмет адрес из своего кэша, не обращаясь к удаленному серверу. Более подробно о DNS мы поговорим в *главе 11*.

С развитием Интернета и потребностей пользователей стали расти и возможности. Прокси-сервер может играть роль шлюза и без дополнительных программ или оборудования обеспечить доступ в Интернет. Помимо этого, он становится щитом в сети от вторжения извне. Например, если все пользователи подключены к Интернету через прокси-сервер, а он прячет реальный IP-адрес пакетов и отправляет их в сеть от своего имени, то хакеры увидят только IP-адрес прокси-сервера и будут ломать его, а компьютеры реальных пользователей останутся незатронутыми. Таким образом, при организации защиты от внешнего вторжения можно больше внимания уделять охране прокси-сервера, чем клиентских компьютеров.

Но, несмотря на возможности прокси-серверов с точки зрения защиты, они слишком просты 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 просто пересылают пакеты пользователя (без изменения адреса отправителя) дальше на WEB-сервер. Прокси-сервер, который скрывает IP-адрес, называется анонимным. Такой сервер общается с внешним миром от своего имени. Этим очень часто пользуются злоумышленники. Например, если хакер хочет вскрыть сервер и замести следы, то он производит все свои действия через анонимный прокси-сервер, чтобы администратор не смог узнать, кто именно производил взлом.

На данный момент в Интернете работает множество анонимных прокси-серверов, но не все из них реально прячут адрес. В отдельных случаях источник остается доступным для удаленной системы, а некоторые серверы сохраняют всю активность в журналах, и их могут просмотреть правоохранительные органы. Таким образом, злоумышленник не может быть уверенным, что используемый им сервер действительно анонимен.

У публичных прокси-серверов Интернета есть еще один недостаток — нельзя гарантировать, что на нем нет никакой заразы. Например, месяц назад у нас на работе администратор запретил доступ к сайтам **vkontakte.ru** и **odnoklasniki.ru**. Я думаю, многие из вас войдут в мое положение и поймут мое горе. Этот запрет легко обходится с помощью прокси, но проблема в том, что первый же публичный сервер, который я нашел, одарил меня целой кучей вирусов и программ-шпионов.

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 прокси есть очень удобная возможность — обмен информацией между серверами. Например, в здании в одной большой сети находятся несколько офисов, каждый из них платит за Интернет отдельно, но общается с внешним миром через свой прокси-сервер. Можно объединить несколько прокси-серверов, и если на одном нет в кэше нужного сайта, то он попытается взять информацию с соседнего сервера.

Чаще всего для реализации такой возможности применяется протокол ICP (Internet Cache Protocol, протокол интернет-кэширования). Если ваш сервер не нашел нужного документа, то он направляет ICP-запрос другим серверам. Если какой-либо прокси-сервер ответит положительно, то информация будет взята у него.

При использовании протокола ICP (или иного способа поиска данных в других прокси-серверах) выигрыш в скорости загрузки не столь заметен при обращении

к документам маленького размера, потому что возрастают расходы времени на ICP и поиск информации в кэше. При большой нагрузке на серверы и большой базе кэша поиск может оказаться слишком долгим, и преимущество в скорости исчезает. Единственное, чем полезно такое решение — экономия трафика, которая может сберечь деньги тем, кто оплачивает каждый получаемый мегабайт.

Мы рассмотрели основные возможности прокси-серверов, но это не значит, что все они есть в любом сервере. Все зависит от разработчика, и некоторые реализуют только одну задачу.

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).

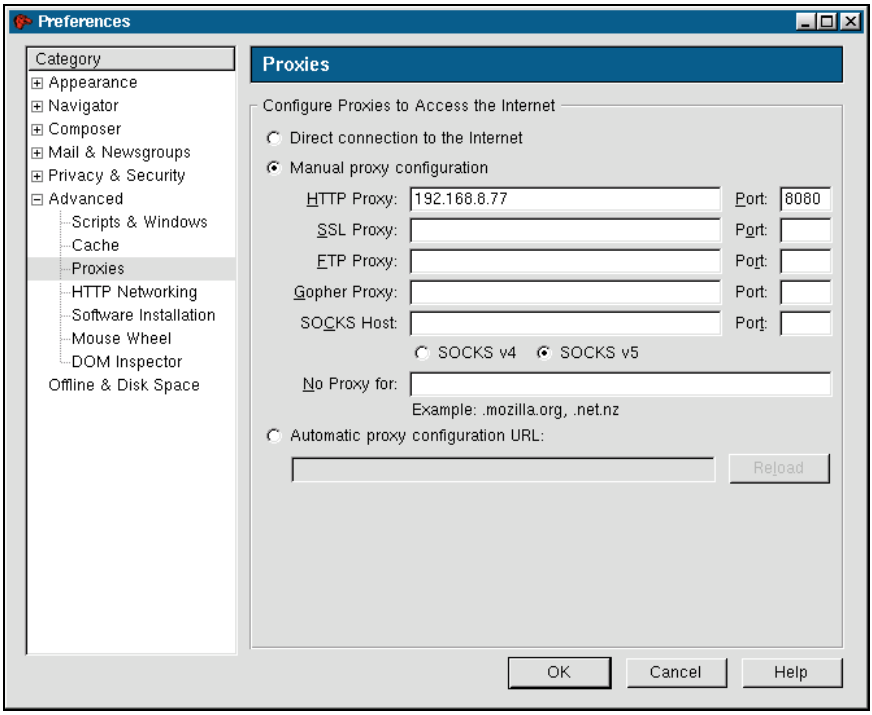


Рис. 9.2. Настройка соединения через прокси-сервер в браузере Mozilla

После этого браузер будет посылать все запросы прокси-серверу, а тот уже перенаправит их серверу, если не может ответить самостоятельно. Прокси-сервер должен постоянно 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 протокола HTTP, применяемого для загрузки WEB-страниц, чаще всего используются порты 3128 или 8080, но эти значения зависят от сервера и могут быть изменены. Перед использованием определенной программы прокси-сервера убедитесь, что она обладает необходимыми вам возможностями и обеспечивает поддержку всех нужных протоколов. Неподдерживаемые протоколы придется направлять не через сервер, а напрямую, через шлюз.

Для повышения безопасности вашей сети необходимо с помощью сетевого экрана запретить подключения извне на используемые сервисом squid порты. Например, для работы с протоколом HTTP по умолчанию им 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.

Основной конфигурационный файл для squid — /etc/squid/squid.conf (в некоторых системах место его расположения /etc/squid.conf). Файл очень велик, и приводить его полностью нет смысла, так как значительную его часть занимают подробные комментарии по использованию директив.

Рассмотрим основные команды, которые доступны нам для управления прокси-сервером. Как обычно, все параметры, влияющие на производительность и безопасность, мы разберем подробно. Остальные настройки будут рассмотрены поверхностно, с ними можно поближе познакомиться, прочитав комментарии из конфигурационного файла.

9.3.1. Директивы настройки HTTP

При подключении к Интернету пользователи первым делом загружают WEB-страницы. Если используется squid, то необходимо правильно настроить

протокол HTTP. Для решения этой задачи в файле squid.conf есть следующие директивы:

- `http_port n` — номер порта, через который будет происходить подключение. Порты, на которых сервер будет ожидать подключения клиентов, — это первое, что необходимо настроить. Такие директивы имеют формат `XXXX_port`. Для порта HTTP запись будет выглядеть таким образом:

  ```
  http_port 8080
  ```

  При конфигурировании браузера на клиентском компьютере вы должны будете указать IP-адрес сервера, где установлен squid, и прописанный в данной директиве порт;

- `hierarchy_stoplist` — определяет перечень URL-адресов, данные с которых всегда должны получаться с сервера, а не из кэша. Я рекомендую добавить в этот список слова "cgi-bin" и вопросительный знак. Адреса URL, содержащие такой текст, указывают на сценарии, которые могут исполняться на сервере, и их результат желательно не кэшировать.

  Рассмотрим пример. Предположим, что вы прочитали WEB-страницу **www.servername.com/cgi-bin/ping.cgi**, на которой можно через WEB-интерфейс выполнить команду `ping`. Допустим, что при первом обращении вы запустили команду `ping` к адресу `18.1.1.1`. Результат будет сохранен в кэше прокси-сервера. В следующий раз вы обращаетесь к сценарию, чтобы выполнить `ping 18.1.1.18`, но браузер вернет первый результат, потому что возьмет его из своего кэша.

  Страницы со сценариями могут возвращать разный результат, в зависимости от ситуации и параметров, которые выбрал пользователь. Если кэшировать такие страницы, то вы всегда будете видеть одно и то же. В результате вы получите только неудобства от соединения через прокси-сервер.

  Вопросительный знак очень часто используется для передачи параметров, 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
  ```

  Думаю, вы согласитесь со мной, что незачем кэшировать то, что все равно при новом запросе будет получаться с сервера, и зря расходовать дисковое пространство.

9.3.2. Директивы настройки FTP

Для работы по протоколу FTP тоже есть несколько директив:

- `ftp_passive параметр` — режим работы. Если в качестве параметра указано значении `on` (устанавливается по умолчанию), то пассивный режим разрешен.

  Сервер squid позволяет работать с протоколом FTP, но для этого может потребоваться дополнительная настройка. Например, если squid находится за сетевым экраном, запрещающим пассивный режим, то лучше изменить значение параметра по умолчанию, установив для этого следующую директиву:

  ```
  ftp_passive off
  ```

- `ftp_user адрес` — определяет адрес E-mail, который будет использоваться в качестве пароля при авторизации на анонимном FTP-сервере.

  Ни один сервер не может точно сказать, правильно ли вы указали адрес, поэтому проверка может быть отключена. Но некоторые FTP-серверы проверяют корректность написания адреса. По умолчанию squid использует в качестве E-mail слово `squid@`:

  ```
  ftp_user squid@
  ```

  По умолчанию в файле /etc/squid/squid.conf эта строка закомментирована, но желательно поменять в ней адрес E-mail, например:

  ```
  ftp_user squid@hotmail.com
  ```

  Такой адрес любой FTP-сервер воспримет как корректный, потому что он соответствует всем правилам написания E-mail;

- `ftp_list_width n` — число `n` задает ширину листинга при просмотре содержимого FTP-сервера. Это значение должно быть достаточным, чтобы увидеть все файлы. Если установить слишком маленькое значение, то имена файлов будут обрезаться.

9.3.3. Настройка кэша

От того, как вы настроите кэш, зависит удобство работы через прокси-сервер, поэтому я постараюсь показать все соответствующие директивы и подробно рассмотреть каждую из них:

- `cache_dir тип 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 это может вызвать проблемы в работе), то нужно установить `aufs`.

Директорию нужно выбрать 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 Мбайт. Этого достаточно для ускорения работы трех пользователей. Если в вашей сети много пользователей, и у каждого свои вкусы (любимые сайты), то значение желательно увеличить. Я использую не менее 1 Гбайт кэша. Выделенное пространство быстро исчезает, особенно если серверу разрешено кэшировать большие файлы;

- `cache_mem n MB` — задает максимальный размер оперативной памяти, необходимый для программы. По умолчанию используется 8 Мбайт. Если настраиваемый компьютер решает только задачи прокси-сервера, то можно указать значение, равное разнице объемов оперативной памяти и памяти, необходимой для ОС. Для ОС в текстовом режиме 64 Мбайт будет более чем достаточно, и, например, если у вас ОЗУ 512 Мбайт, то 448 Мбайт можно отдать прокси-серверу: чем больше у него оперативной памяти, тем быстрее он сможет отвечать на часто повторяемые запросы;
- `cache_swap_low n` — процент заполнения кэша. Когда размер кэша превышает значение `n`, сервер начинает 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, который попадает 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 Мбайт. Для повышения производительности сервера необходимо понизить это значение, но при этом может увеличиться расход трафика. Если экономия трафика важнее производительности, то значение `n` лучше увеличить;
- `maximum_object_size_in_memory n KB` — максимальный размер объекта в памяти. По умолчанию установлено значение 8 Кбайт;
- `ipcache_size n` — размер кэша для хранения IP-адресов. По умолчанию 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 year
  ```

- `quick_abort_min n KB` — минимальный размер объекта, устанавливающий при обрыве соединения необходимость закончить его скачивание и полностью сохранить. Это позволяет сократить трафик и увеличить скорость работы в сети. Например, пользователь запустил на скачивание файл для проверки соединения и оборвал связь. Если сервер успел сохранить файл, то при повторной попытке не надо снова скачивать те же данные. Достаточно их взять из кэша. По умолчанию установлено значение `16`. Чтобы отключить эту возможность, можно задать значение `-1`;

- `quick_abort_max n KB` — максимальный остаток объекта, при котором закачка будет прервана в случае обрыва соединения. По умолчанию установлено значение `16`;

- `quick_abort_pct n` — параметр аналогичен `quick_abort_min`, но в данном случае указывается процент уже полученной информации;

- `negative_ttl n minutes` — количество минут, которые нужно кэшировать негативный ответ сервера. Например, пользователь зашел на сервер и получил ошибку, которая может быть временной, поэтому нельзя кэшировать ответ на длительный срок. Значение по умолчанию 5 минут. Если пользователь обратится по этому же адресу по истечении этого времени, то копия из кэша не будет использоваться, а произойдет новая попытка зайти на сайт;

- `positive_dns_ttl n hours` — время в часах, в течение которого нужно кэшировать положительный результат DNS-запроса. В этот промежуток времени при повторных обращениях к DNS IP-адрес будет взят из кэша. По умолчанию 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 статический 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 по имени не найден IP-адрес, то, возможно, проблема с сервером имен, а не с именем. Такие вопросы чаще всего решаются в течение 2—3 минут, поэтому отрицательный ответ не стоит держать в кэше дольше, иначе все это время клиенты не смогут обратиться к сайту. Я делаю этот параметр равным 1 или 0, чтобы пользователи увидели нужный сайт сразу после устранения проблемы;
- `range_offset_limit n KB` — порядок передачи клиенту неполных данных. Пользователь может запросить не весь файл, а лишь его часть. В этом случае squid не может кэшировать такой файл, не скачав его целиком. Однако если запрошенная часть сильно смещена от начала файла, то это приведет к большой задержке перед тем, как squid начнет возвращать что-либо клиенту. Для контроля над такими ситуациями и предусмотрен этот параметр: если смещение запрошенной части более параметра `n`, то прокси-сервер запросит у сервера в Интернете лишь необходимую пользователю часть файла и не будет его кэшировать. Если указать значение `-1`, то squid будет всегда загружать весь файл и кэшировать его содержимое. Если же указать значение `0` (по умолчанию), то, напротив, squid будет скачивать лишь запрошенные части файлов и кэшировать их, лишь если запрошен весь файл.

9.3.4. Журналы

В конфигурационном файле есть несколько параметров, влияющих на работу прокси-сервера с журналом (журналы легко читаются в любом текстовом редакторе):

- `cache_access_log файл` — журнал, в котором сохраняется вся активность пользователей, а именно, HTTP- и ICP-запросы. По умолчанию этот параметр равен /var/log/squid/access.log;
- `cache_log файл` — файл для хранения основной информации о кэше. По умолчанию 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 утилит для анализа сохраняе-

мых данных нет, да и вообще пользы в таких данных мало, а расходы на их сохранение присутствуют;

- `log_mime_hdrs параметр` — если в качестве параметра указано `on`, то в журнале access будут сохраняться заголовки MIME;
- `useragent_log` — журнал, в котором сохраняется поле `User-agent` заголовков HTTP. Смысла в этом поле нет, потому что его легко подделать, и ничего полезного журнал не даст, поэтому по умолчанию он не используется.

В *разд. 12.5* мы будем говорить о журналах различных сервисов Linux, а в *разд. 12.5.4* мы 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 используется 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 используется 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 протокол ICP, 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 утяжелять книгу. В комментариях, содержащихся в конфигурационном файле, каждый параметр подробно описан;

- `icp_query_timeout n` — время ожидания в миллисекундах. Чаще всего прокси-серверы расположены в локальной сети с высокой скоростью доступа, и ожидание более 2000 мс будет лишним. Иначе, если ответ не будет получен и придется обращаться в Интернет, пользователь ощутит большую задержку;
- `cache_peer_domain хост домен` — разрешить для соседнего прокси-сервера, расположенного по адресу `хост`, работу только с указанными доменами. Например, следующая строка позволит обращаться к соседнему прокси-серверу с адресом 192.168.2.20 (который должен быть описан директивой `cache_peer`) только за тем, что относится к домену **.com**:

  ```
  cache_peer_domain 192.168.2.20 .com
  ```

  Все остальные запросы не будут направляться на указанный соседний сервер, чтобы не перегружать его лишней работой. С помощью этого параметра можно настроить в сети несколько прокси-серверов, где каждый будет отвечать за свой домен.

9.3.6. Дополнительные директивы

Рассмотрим оставшиеся представляющие для нас ценность параметры, которые я не смог отнести к определенным категориям:

- `redirect_rewrites_host_header параметр` — разрешение (`on`) или запрет (`off`) на изменение поля `Host` в заголовках запросов. Если изменение разрешено, то сервер работает в анонимном режиме, иначе — в прозрачном. Анонимный режим требует дополнительных затрат, но позволяет использовать всего один IP-адрес для внешних соединений в сети любого размера. Прозрачный режим работает быстрее, но каждый компьютер должен иметь собственный IP-адрес, подходящий для работы с Интернетом;
- `redirector_access список` — перечень запросов, проходящих через перенаправитель (redirector). Перенаправитель — это небольшая программа, которая просматривает запрашиваемые URL и, возможно, заменяет их. Этот механизм позволяет реализовать довольно эффективный фильтр, запрещающий, например, просмотр порнографии. Более подробную информацию можно найти в *разд. 9.5.6* и в документации по squid. По умолчанию перенаправителю отправляются все запросы;
- `cache_mgr email` — адрес E-mail, на который будет послано письмо в случае возникновения проблем с работой прокси-сервера;
- `append_domain домен` — домен по умолчанию. Например, чаще всего пользователи работают с адресами домена **.com**. Вполне логичным будет указать в директиве именно его (`append_domain .com`). Если пользователь введет адрес redhat, то squid сам добавит имя домена, и направит на сайт **redhat.com**;

- `smtp_port n` — номер порта, на котором нужно ожидать SMTP-запросы для отправки сообщений. Конечно же, SMTP — это такой протокол, который не требует кэширования, и работа через прокси-сервер не сэкономит трафик, но возможность может оказаться полезной, если нельзя устанавливать шлюз, а разрешен только прокси-сервер;
- `offline_mode параметр` — режим работы. Если параметр равен `on`, то squid будет взаимодействовать только с кэшем, и обращений к Интернету не будет. Если запрашиваемой страницы в кэше нет, то пользователь увидит ошибку. Чтобы squid мог обращаться к Интернету, необходимо установить параметр `off` (установлено по умолчанию).

9.4. Права доступа к squid

Это самая 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 директиву:

```
acl имя тип параметр
```

У данной директивы три параметра:

- `имя` — может быть любым, но лучше всего, если оно будет описывать выполняемые действия;
- `параметр` — задает шаблон или строку, смысл которой зависит от типа записи (второй аргумент);
- `тип` — может принимать следующие значения: `src`, `dst`, `srcdomain`, `dstdomain`, `url_pattern`, `urlpath_pattern`, `time`, `port`, `proto`, `proxy_auth`, `method`, `browser`, `user`. Рассмотрим основные типы, которые вам пригодятся при формировании последнего параметра (шаблона):
  - `src` — задаются IP-адреса пользователей;
  - `dst` — указываются адреса серверов;

- `port` — определяется номер порта;
- `proto` — описывается список протоколов (через пробел);
- `method` — указывается используемый метод, например `POST`, `GET` и т. д.;
- `proxy_auth` — определяется список имен пользователей, значения разделяются пробелами. В качестве имени можно использовать `REQUIRED`, чтобы принимались любые действительные для сервера имена (`acl password proxy_auth REQUIRED`);
- `url_regex` — устанавливается URL или регулярное выражение для URL;
- `time` — задается время в формате `дни h1:m1-h2:m2`. С помощью такой записи можно ограничить доступ только определенными днями недели и обусловленным временем. В качестве дней недели можно указывать: `S` — Sunday (воскресенье), `M` — Monday (понедельник), `T` — Tuesday (вторник), `W` — Wednesday (среда), `H` — Thursday (четверг), `F` — Friday (пятница), `A` — Saturday (суббота).

В файле конфигурации уже описано несколько правил, которые готовы к использованию и в большинстве случаев не требуют вмешательства. 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 работы прокси-сервера.

Рассмотрим первую строку. Здесь задается ACL с именем `all`. Так как используется тип шаблона `src`, то этому списку принадлежат пользователи, у которых IP-адрес соответствует 0.0.0.0/0.0.0.0, то есть все пользователи.

Следующая строка создает ACL с именем `manager`. Она определяет доступ к протоколу, потому что тип записи `proto`, а последний параметр задает протокол — `cache_object`. И так далее.

Давайте попробуем задать свою запись ACL. Допустим, в вашей сети есть 10 компьютеров с адресами от 192.168.8.1 до 192.168.8.10 (маска подсети 255.255.255.0), которым разрешен доступ к Интернету. Значит, всем остальным нужно запретить.

Уже при создании списка вы должны отталкиваться от идеи, что изначально доступ запрещен всем, и позволять 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 `имя` — это имя ACL-записи. В следующем примере запрещается доступ по протоколу HTTP всем пользователям, кроме указанных в ACL-записи `AllowUsers`:

  ```
  http_access deny all
  http_access allow AllowUsers
  ```

  Указав права доступа для списка `AllowUsers`, мы одной строкой даем разрешение для всех компьютеров, входящих в 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 описали список `AllowUsers` как список компьютеров с IP-адресами из диапазона 192.168.8.1—192.168.8.10. Если к прокси-серверу обратится компьютер с другим адресом, то в доступе будет отказано;

- `icp_access разрешение имя` — описывает разрешение доступа к прокси-серверу по протоколу ICP. По умолчанию доступ запрещен для всех:
  ```
  icp_access deny all
  ```
- `miss_access разрешение имя` — описывает разрешение на получение ответа MISSES. В следующем примере только локальным пользователям дано право получать ответ MISSES, а все остальные могут принимать только HITS:
  ```
  acl localclients src 172.16.0.0/16
  miss_access allow localclients
  miss_access deny  !localclients
  ```

9.4.3. Аутентификация

Защита по IP-адресу не гарантирует от его подделки злоумышленником. К тому же, остается вероятность, что кто-то получит физический доступ к компьютеру, имеющему доступ во всемирную сеть, и сделает нечто неразрешенное.

Мне довелось работать в фирме, где каждому сотруднику было разрешено скачивать из сети определенное количество мегабайт. Проверка проходила через IP-адрес, и при превышении лимита руководство требовало от работника покрыть расходы за сверхнормативный трафик. Это нормальная ситуация, потому что работодатель не должен оплачивать бессмысленные прогулки сотрудника в Интернете. На работу приходят, чтобы выполнять свои обязанности, а не подыскивать обои для собственного компьютера.

Однажды у некоторых работников оказалось большое превышение трафика. Вроде бы ничего удивительного, но настораживало то, что эти товарищи были в 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 используется) и файл паролей. Если вы хотите использовать традиционную программу аутентификации, то можно указать следующую строку:
  ```
  authenticate_program /usr/lib/squid/ncsa_auth /usr/etc/passwd
  ```
  Путь к программе ncsa_auth в вашей системе может 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 может производить проверку нескольких клиентов, поэтому если один пользователь проходит аутентификацию, то другие не смогут получить доступ к Интернету через сервер squid;
- `authenticate_ttl n hour` — срок (в часах) хранения в кэше результата аутентификации. В течение этого времени пользователь может работать без повторной проверки. По умолчанию установлен 1 час, но если введен неправильный пароль, то запись удаляется из кэша;
- `authenticate_ip_ttl 0 second` — связывает IP-адрес с аутентификацией. Необходимо установить `0`, чтобы пользователи не могли воспользоваться одним и тем же паролем с разных IP-адресов. Некоторые пользователи считают, что можно делиться паролем с друзьями, но не стоит им это разрешать, потому что за раздачу паролей должен отвечать только администратор.

  Если в вашей сети есть dialup-пользователи, подключающиеся с помощью модема, то это значение можно увеличить до 60 секунд, чтобы при разрыве связи была возможность перезвонить. Но обычно при dialup-подключении используются динамические IP-адреса, которые выдаются при каждом соединении, и нет гарантии, что после повторного звонка адрес сохранится;
- `authenticate_ip_ttl_is_strict параметр` — если параметр равен `on`, то доступ с других IP-адресов запрещен, пока время, указанное в `authenticate_ip_ttl`, не выйдет.

> ***Внимание!***
>
> Аутентификация не работает, если squid настроен на работу в прозрачном режиме.

9.5. 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 поль-

зователя и группы будут заменены на указанные в этих параметрах. По умолчанию установлено значение идентификатора squid и для пользователя, и для группы:

```
cache_effective_user squid
cache_effective_group squid
```

Таким образом, squid не будет работать с правами root, и при попытке сделать это сервис сам понизит свои права до squid. Не стоит вмешиваться. Сервису squid не имеет смысла давать большего, потому что ему достаточно прав только на работу с директорией кэша.

9.5.2. Ускорение сайта

Сервис squid может ускорить работу определенного сайта, функционируя как httpd-акселератор. Для этого необходимо указать как минимум три параметра: адрес ускоряемого сервера, который надо кэшировать, его порт и атрибуты сервера, который надо ускорять. Это делается с помощью следующих директив:

- `httpd_accel_host адрес` — адрес реального сервера;
- `httpd_accel_port порт` — порт WEB-сервера. Чаще всего это порт по умолчанию (он равен 80), если не указан другой;
- `httpd_accel_uses_host_header параметр` — заголовок HTTP включает в себя поле Host. Сервер squid не проверяет Host, и это может оказаться большой дырой в безопасности. Разработчики рекомендуют отключать эту директиву, указав в качестве параметра значение `off`. Включать ее необходимо, если squid работает в прозрачном режиме;
- `httpd_accel_with_proxy параметр` — флаг использования кэширования страницы для дополнительного повышения скорости (`on`/`off`).

9.5.3. Маленький секрет User Agent

Многие статистические системы не учитывают или не пускают к себе пользователей, запросы которых содержат пустое значение в поле User Agent. Именно так определяется, что вы работаете через прокси-сервер.

Опять случай из собственной практики. Я снова вспоминаю фирму, где мне пришлось работать 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 прокси-серверу. Так как все запросы шли от меня, то главный прокси-сервер не возражал.

На первый взгляд решение идеальное, но тут есть один изъян. Да, это поле User Agent, которое становится пустым при прохождении пакетов через мой squid-сервис. Но поле можно задать вручную в конфигурационном файле. Для этого существует директива `fake_user_agent`. Например, следующая строка может эмулировать запросы, как будто они идут от браузера Netscape:

```
fake_user_agent Netscape/1.0 (CP/M; 8-bit)
```

9.5.4. Защита сети

Сервис squid может быть как средством защиты сети, так и орудием проникновения хакера в сеть. Чтобы внешние пользователи не могли задействовать прокси-сервер для подключения к компьютерам локальной сети, необходимо добавить в конфигурационный файл следующие строки:

```
tcp_incoming_address внутренний_адрес
tcp_outgiong_address внешний_адрес
udp_incoming_address внутренний_адрес
udp_outgiong_address внешний_адрес
```

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 компьютерам локальной сети, находясь за ее пределами. Вот пример ошибочного конфигурирования squid-сервиса:

```
tcp_incoming_address внешний_адрес
tcp_outgiong_address внутренний_адрес
udp_incoming_address внешний_адрес
udp_outgiong_address внутренний_адрес
```

9.5.5. Борьба с баннерами и всплывающими окнами

В фирме, где я работал, появился новый сотрудник, и в первую неделю мы ощутили увеличение трафика. Это бывает со всеми, потому что любой новый пользователь Интернета начинает смотреть все страницы подряд. Со временем интерес стихает, и трафик понижается.

Мы уже говорили о том, что на любом сайте бóльшую 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 информацию. Если найден ошибочный путь, то его можно записать в этот файл, и баннер будет загружен.

Теперь добавляем еще две строки после описания ACL-записей:

```
http_access deny banners_path_regex !banners_exclusion
http_access deny banners_regex !banners_exclusion
```

Обе директивы имеют один и тот же смысл — запрещается загрузка по адресам, прописанным в списке `banners_path_regex` или `banners_regex`, если адрес не входит в исключение, описанное в файле ACL-списка `banners_exclusion`.

Рассмотрим фрагмент содержимого файла /usr/etc/banners_regex:

```
^http://members\.tripod\.com/adm/popup/.+html
^http://www\.geocities\.com/ad_container/pop\.html
```

Напоминаю, что в этом файле находятся URL-пути для сравнения, и все адреса, которые им соответствуют, будут отфильтрованы.

Шаблону из первой строки соответствует, например, адрес **http://members.tripod.com/adm/popup/popup.html**, так что он не будет загружен. 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 такая большая, что 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, и позволяют 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 перенаправитель (redirector). Для сервиса squid это внешняя программа, которая подменяет адреса. Например, если сайту необходимо загрузить баннер, и ваша программа смогла определить такую попытку, то redirector подменит адрес и вместо баннера загрузит то, что укажете вы.

Есть только одна проблема — в ОС нет, и не может быть 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 — файл HTML, который будет закрывать всплывающие окна. В нем нужно поместить всего лишь функцию, которая будет закрывать окно. Для этого используется функция JavaScript `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 наиболее высок (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 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 "Ga**sEx**po". Обратите внимание, что выделенные буквы создают слово "sex". И случаи, когда пользователи не могут попасть на сайт выставки по газовому оборудованию, вполне реальны.

Создавать списки запрещенных сайтов достаточно сложно, но можно. В настоящее время в зоне **com** большинство сайтов эротической направленности закрылись, и они обживают другие домены, принадлежащие маленьким государствам. Существуют домены, которые на 90% состоят из сайтов индустрии развлечений для взрослых. Вот их можно обрезать полностью.

9.5.8. Ограничение канала

При организации доступа в Интернет очень часто требуется обеспечить отдельным пользователям бóльшую скорость подключения. Как это сделать, когда по умолчанию все равноправны и могут работать на максимально доступной на данный момент скорости? Для этого нужно определиться с приоритетами. Для некоторых пользователей должен быть зарезервирован высо-

коскоростной 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 три строки, поэтому в первой параметр `n` равен `1`, во второй — `2`, а в третьей — `3`.

Номеров класса (второй параметр) может быть три:

- 1 — ограничение канала происходит для всей сети. Например, у вас внешний канал 256 Кбит/с, вы можете его уменьшить для всех до 64 Кбит/с;
- 2 — сузить можно общий канал, а также для каждого пользователя индивидуально. В этом случае можно понизить общий канал до 64 Кбит/с, а каждому пользователю — до 4 Кбит/с;
- 3 — ограничивать можно общий канал, индивидуально и для каждой сети в отдельности. Например, если скорость канала равна 256 Кбит/с, а в вашей сети работает 4 подсети, то каждой из них можно выделить по 64 Кбит/с, чтобы равномерно разделить нагрузку.

В нашем примере мы используем пулы классов 1, 2 и снова 1. Я специально расположил их не последовательно, чтобы пример был более наглядным.

Теперь описываем параметры скорости доступа. Это делается с помощью следующей директивы:

```
delay_parameters пул скорость_канала скорость_сети индивидуально
```

Здесь `пул` — это номер пула, скорость которого мы хотим описать. Так, в нашем примере следующая строка описывает скорость для первого пула:

```
delay_parameters 1 256000/256000
```

Количество параметров зависит от класса используемого пула. Так как пул 1 имеет класс 1 (`delay_class 1 1`), у которого можно ограничивать только канал полностью, в директиве используется единственный параметр — `скорость_канала` (`256000/256000`). Он формируется в виде двух чисел, разделенных знаком "/". До косой черты указывается скорость, с которой будут скачиваться данные из сети, а после — размер пула, то есть емкость, которую можно наполнить полученными из сети данными.

Обратите внимание, что скорость указывается именно в байтах, а характеристика модема задается в битах в секунду. Если в качестве скорости указать значение `-1`, то никаких ограничений не будет.

После определения параметров для первого пула нужно установить права доступа к нему. Это делается директивой `delay_access`, которая имеет следующий вид:

```
delay_access пул доступ acl
```

Первый параметр — это снова номер пула. Потом указывается доступ `allow` или `deny`, и последним идет имя ACL-списка.

В нашем примере для первого пула 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 мультимедиа-файлов в рабочее время. В следующем примере мы разрешаем быстро читать WEB-странички, но уменьшаем скорость загрузки других медиа-файлов (см. листинг 9.4).

Листинг 9.4. Ограничение скорости загрузки медиа-файлов в рабочее время

```
# ACL-список, описывающий сеть
acl fullspeed url_regex -i 192.168.1
# ACL-список, описывающий медиа-файлы, 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 медиа-файлов
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 пропорционально их числу, а медленнее.

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, используемой для хранения последних загруженных страниц. Использование кэша в памяти увеличивает скорость, когда вы путешествуете по одному и тому же сайту, ведь 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.

Другой 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 папка для хранения кэша.

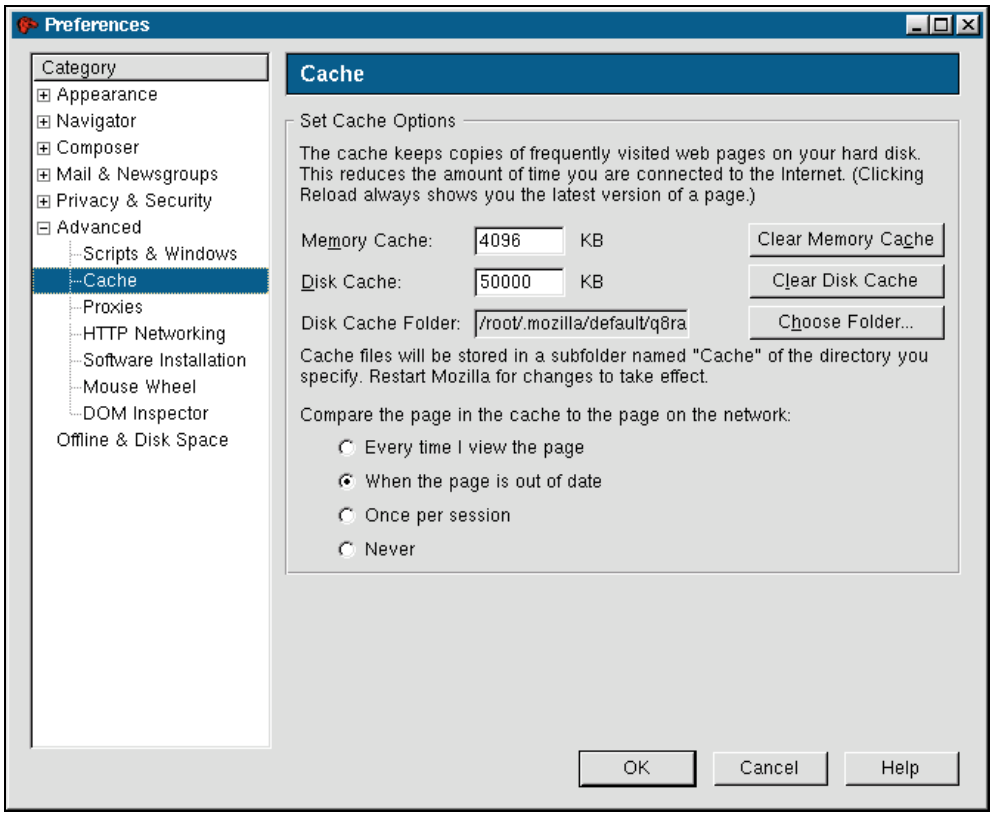


Рис. 9.3. Настройка кэширования в браузере Mozilla

И последнее, что можно настроить — условия, 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, то есть локальный кэш использоваться не будет;
- **When the page is out of date** — когда страница устарела;
- **Once per session** — единожды за сессию, то есть только при первом заходе;

- **Never** — никогда. Страница всегда будет загружаться из кэша, а для обновления нужно нажать кнопку **Reload** (Обновить) 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**.

9.7. squidGuard

Использование прокси-сервера достаточно опасно, поэтому squid очень часто используют совместно с дополнительной защитой в виде программы squidGuard. В качестве возможностей этого стражника можно выделить опознавание пользователей по имени или адресу, фильтрацию запросов, замену баннеров и т. д.

9.7.1. Установка

Установка программы достаточно интересна. Скачайте исходные коды с сайта **www.squidguard.org**. Скачав исходники, разархивируйте их. Это можно сделать даже графической утилитой, которая работает в Ubuntu как файловый менеджер. Для простоты разархивируйте их в свою домашнюю директорию. Запустите терминал и перейдите внутрь созданной при разархивации директории с программой:

```
cd ~/squidGuard-X.X
```

В данном случае X.X — это номер версии squidGuard. Запустите отсюда команду конфигурации:

```
./configure
```

По экрану побегут строки, сообщающие о ходе проверки и конфигурирования программы. Если выполнение прервалось с ошибкой, то, вероятнее всего, в вашей системе отсутствует BerkeleyDB. У меня в Ubuntu так и было, поэтому сразу расскажу, как станцевать танец с бубнами для установки этого зверя. При помощи поисковика находим архив BerkeleyDB, Скачиваем его и разархивируем. Лучше опять разархивировать в домашнюю папку, все равно потом все это можно будет удалить. Теперь переходим в папку:

```
cd ~/db-X.X.X/build-unix
```

Буквы в виде X снова означают номер версии. Обратите внимание, что мы переходим в подкаталог build-unix. Это очень важно. Мы устанавливаем библиотеку работы с базой в Linux, который относится к системам UNIX, поэтому

компиляция и установка должна происходить именно отсюда. В этой директории выполняем три команды:

```
../dist/configure
make
sudo make install
```

Первая команда конфигурирует программу, вторая ее компилирует, а третья — устанавливает. Обратите внимание, что перед командой установки стоит sudo. Дело в том, что установка будет происходить в директорию /usr/local, к которой у простых смертных учетных записей доступа нет. Если не указать sudo, то установка завершится ошибкой прав доступа.

Теперь, когда библиотека базы данных установлена, можно возвращаться в директорию squidGuard. Только не спешите запускать конфигурацию снова. С вероятностью 99% она не пройдет, потому что конфигуратор ищет библиотеку BerkeleyDB в папке /usr/local/BerkeleyDB, но она устанавливается в /usr/local/BerkeleyDB.X.X, то есть в конец имени папки добавляется номер версии. Выполните команду:

```
ls /usr/local
```

Это позволит увидеть содержимое папки /usr/local прямо из текущей папки стражника. Запомните имя и выполняйте команду конфигурации так:

```
./configure --with-db=/usr/local/BerkeleyDB.X.X
```

В данном случае X.X на конце заменяем на свой номер версии. Теперь конфигурация должна пройти успешно. Если нет, то в этом случае все претензии к разработчикам, но я с другими проблемами не встречался.

После конфигурации выполняем команды сборки и установки:

```
make
sudo make install
```

Установка снова выполняется от имени администратора, потому что снова происходит копирование в защищенную область /usr/local.

9.7.2. Настройка

Директория по умолчанию, в которой можно найти установленный нами стражник — /usr/local/squidGuard. Основные настройки находятся в файле squidGuard.conf, расположенном в этой же директории. Первое, что вы увидите в этом файле, это следующие две строки:

```
dbhome /usr/local/squidGuard/db
logdir /usr/local/squidGuard/logs
```

Первый параметр (`dbhome`) задает путь к базам данных, где будут храниться списки доступа. Второй параметр (`logdir`) — это путь к журналу. На сайте

www.squidguard.org в разделе Blacklists можно скачать примеры списков доступа с вредными сайтами. Их нужно разархивировать в директорию баз данных, то есть ту, что указана в параметре `dbhome`.

Теперь мы можем в этом же файле задать права доступа. Причем права доступа могут распределяться даже по времени и по пользователям. Отдельным пользователям можно дать возможность смотреть все, а кому-то (например, детям) запретить смотреть сайты для взрослых.

Для создания времени используется параметр `time`. Например, можно создать период, который будет отображать рабочее время. Для этого в файле squidGuard.conf пишем:

```
time worktime {
 weekly * 9:00-17:00            # Рабочее время
 date 2010.02.27 9:00-17:00     # День, который случайно стал рабочим
}
```

После параметра `time` указывается имя периода. В фигурных скобках указываем периоды времени. Параметр `weekly` со звездочкой означает любой день недели. Если нужно определиться только с рабочими днями недели, то их можно перечислить как MTWHF. После этого идет рабочее время, в которое и будет использоваться данное правило. Параметр `date` позволяет задать определенную дату. В данном случае я выбрал для примера субботу, 27 февраля 2010 года, которая будет рабочим днем. Наше правило будет распространяться и на этот день.

Для задания адресов, к которым будет разрешен или запрещен доступ, используется директива `dest`, например:

```
dest porn {
      domainlist       porn/domains
      urllist          porn/urls
      expressionlist   porn/expressions
 }
```

Как видно из примера, здесь указываются параметры `domainlist`, `urllist` и/или `expressionlist`, с помощью которых задаются имена файлов, содержащих списки доменов, URL или регулярных выражений. Пути задаются относительно директории, указанной в директиве dbhome. Например, файл /usr/local/squidGuard/db/porn/domains может содержать тысячи строк вроде

```
sex.com
sexygirls.com
```

Регулярные выражения позволяют записывать множество сайтов в одну строку, но злоупотреблять ими не следует, так как проверка соответствия может занимать много времени.

Теперь посмотрим, как можно написать правило, которое запретит в рабочее время смотреть сайты для взрослых (porn), рекламу (adv) для экономии трафика и сайты с нелегальными программами (warez) для защиты каналов от перегруза:

```
acl {
        all within worktime {
                pass    !adv !porn !warez all
        }
        else {
                pass all
        }
        default {
                pass    none
                redirect http://localhost/block.html
        }
}
```

Это разрешающее правило acl (Access Control List). В нем три блока:

- all within worktime — для всех в рабочее время — разрешено все, кроме перечисленных трех запретов, о которых мы говорили выше. На то, что они запрещены, указывает восклицательный знак перед их названиями;
- else — все остальное время разрешено все;
- default — по умолчанию — вообще ничего (none) не разрешено, а происходит переадресация на **http://localhost/block.html**.

Разумеется, чтобы это правило работало, нужно еще описать списки адресов adv и warez при помощи директивы dest, подобно тому, как был описан porn.

Таким образом мы написали правило, которое касается всех. А что, если вы хотите указать исключение для себя? Администраторы — привилегированные люди, и глупо ограничивать самого себя. Для этого можно прописать группу, например admins, и прописать туда адреса компьютеров, которые должны быть привилегированными, и обрабатывать их отдельно:

```
src admins {
        ip      192.168.1.1-192.168.1.3
}
src others {
        ip      192.168.1.4-192.168.1.255
}
```

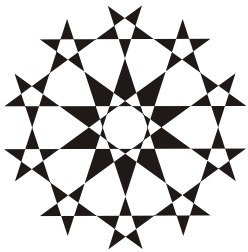
Теперь права доступа можно подкорректировать следующим образом:

```
acl {
        admins within worktime {
                pass all
        }
        others within worktime {
                pass    !adv !porn !warez all
        }
        else {
                pass all
        }
        default {
                pass    none
                redirect http://localhost/block.html
        }
}
```

Теперь вы сможете обращаться к любым WEB-сайтам. Чтобы изменения вступили в силу, нужно выполнить следующую команду:

```
squidGuard -C all
```

Ее выполняем из директории /usr/local/bin.



Глава 10

Передача файлов

Были 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-то застал те времена, наверное, он вспоминает их с ужасом. Дискеты постоянно портились, и данные периодически переставали читаться. Благо, если расстояние было небольшое, и можно было повторить пробежку, но когда дистанция большая — потеря очень сильно отражалась на эмоциональном состоянии.

До сих пор в некоторые компьютеры вставляют дисководы для 3,5-дюймовых дискет, видимо, по привычке. Но уже трудно представить себе офис без полноценной сети, и в фирмах, где мне приходилось настраивать серверы, все компьютеры обязательно подключены к локальной сети. В таких случаях уже нет надобности в дисководах, и администраторы их просто вынимают. Если у вас возник вопрос, зачем вытаскивать то, что может пригодиться, значит, вы забыли, что ничего лишнего в компьютере 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. Да, много не унесешь, но все же. Мне известна фирма, которая содержала локальную сеть, оторванную от внешнего мира, и считала себя неприступной. Но секретная информация утекла с помощью простых дискет через все охранные системы, потому что металлоискатели не срабатывали на 3,5-дюймовые дискеты, в силу чего были потеряны рынки сбыта, и 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 протокола FTP

Как мы уже говорили в *главе 6*, для использования протокола FTP требуется две составляющие: клиент и сервер. В качестве клиента, в принципе, 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 которого можно увидеть на рис. 10.1.

Если у вас нет FTP-клиента, то для тестирования протокола можно использовать даже браузер, например, Internet Explorer или Firefox. Для этого в строке URL нужно набрать адрес в формате `ftp://имя:пароль@адрес`.

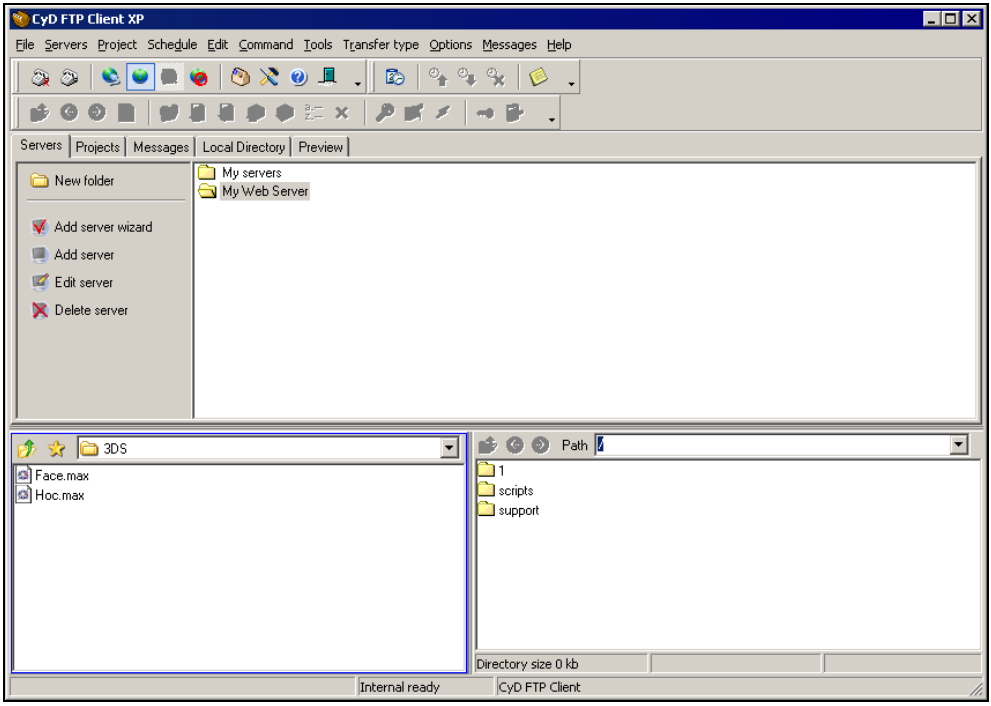


Рис. 10.1. Главное окно программы CyD FTP Client XP

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 портом 21 и начинает передавать команды. С этим портом соединяются все пользователи, и сервис, который прослушивает этот канал, работает одновременно с несколькими соединениями.

Когда клиент запрашивает данные, открывается еще одно соединение с конкретным пользователем, по которому передается файл. Это удобно с точки зрения программирования, но несподручно с точки зрения администрирования, точнее сказать, конфигурирования сетевого экрана. Дело в том, что для передачи данных клиент может открыть абсолютно любой порт, и сервер не может предсказать его номер.

Большинство команд, 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 являлись UNIX-системы. Это в наше время везде стоит Windows, а 30 лет назад все было иначе.

10.1.1. Команды протокола FTP

Давайте рассмотрим листинг 10.1. В нем приведен пример общения клиентской программы с FTP-сервером. Строки, в начале которых стоит знак ">", были отправлены FTP-серверу, а остальные — это его ответы.

Листинг 10.1. Пример работы протокола FTP

```
< 220 Flenov Mikhail FTP Server
> USER Anonymous
< 331 Anonymous access allowed, send identity (e-mail name) as password.
> PASS your@mail.com
< 230 Anonymous user logged in.
> PWD
< 257 "/" is current directory.
> TYPE A
< 200 Type set to A.
> PASV
```

```
< 227 Entering Passive Mode (127,0,0,1,13,20).
> LIST
< 125 Data connection already open; Transfer starting.
< 226 Transfer complete.
```

Самой первой строкой идет приглашение сервера FTP. Его мы получаем сразу же в ответ на соединение с портом 21. В этой строке чаще всего находится текстовое описание сервера, с которым произошло соединение, и его версия. В 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 FTP-сервера, его версия и приветственное сообщение. Ничего страшного не видите? А я вижу — хакеру достаточно подключиться на порт 21, чтобы узнать, с каким FTP-сервером он имеет дело.

Дальнейшие действия взломщика легко предсказать. Я бы на его месте поискал во всех базах уязвимостей информацию о дырах в данной версии wu-ftpd. А администратору остается молиться, чтобы я не нашел нужных сплоитов, или найденная дыра была незначительной и не позволила мне ничего сделать с его системой.

После прочтения строки сообщения можно посылать команды FTP-серверу. Но ничего особого выполнить не удастся, пока вы не представитесь серверу. Для этого нужно выполнить команды FTP: `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 адрес E-mail. Честно говоря, можно ввести и чужой E-mail, например, адрес соседа. 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 программ создают анонимные FTP-серверы для того, чтобы пользователи могли скачать с сервера последние версии программ или обновления для них.

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 закачку файлов на сервер, но в отличие от действительного, он работает только в своей директории. Например, если гостю назначен каталог /home/guest, то он сможет работать с файлами и подкаталогами этой директории, но выше нее подняться не сможет. 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, на заре рождения Интернета никто не думал о хакерах. Теперь приходится выдумывать разные методы, чтобы спрятать пароль.

Если ваш сервер обслуживает только анонимные соединения, то пароли могут передаваться, как угодно. При такой аутентификации любой пользователь и так может подключиться к серверу, указав любой адрес E-mail в качестве пароля. Но подобные серверы 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, о котором мы говорили 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 только 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 пароль, подключиться он 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, описанные в RFC (Requests for Comments, рабочие предложения). Однако возможности, предоставляемые стандартом, уже устарели, и 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, это еще не значит, что она плоха, просто она может быть несовместима с данным сервером.

Основные команды протокола FTP можно увидеть в табл. 10.1. Они вам могут пригодиться при работе через Telnet и для тестирования сервера. Более подробно они описаны в *Приложении 1*.

Таблица 10.1. *Команды протокола FTP*

Команда	Описание
`USER имя`	Авторизоваться на сервере как пользователь с именем `имя`
`PASS пароль`	Указать пароль 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 команду или передачу данных
`QUIT`	Выйти из системы

10.1.2. Сообщения сервера

В ответ на полученную команду сервер возвращает нам сообщения, по которым можно оценить результат работы. Уведомления состоят из трехзначного числового 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, вы можете быстро определить причину сбоя и 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 принята к выполнению, 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. Сообщения с кодом, начинающимся с цифры 3, появляются 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 относится к аутентификации
4	Не определено
5	Сообщение файловой системы

Рассмотрим пример. Допустим, пользователь увидел в своем FTP-клиенте 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, то есть сообщение появилось в ходе аутентификации. 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 увидеть, какая возникла проблема, и 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 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 режиме ASCII. Теперь информация стала читабельной, и все символы перехода на новую строку расставлены самой ОС верно, в соответствии со своими внутренними правилами.

Двоичные файлы (например, картинки или музыку) необходимо передавать в бинарном режиме. Здесь нет различий, в какой ОС создан объект, и он будет правильно воспринят в любой ОС, умеющей работать с данным форматом.

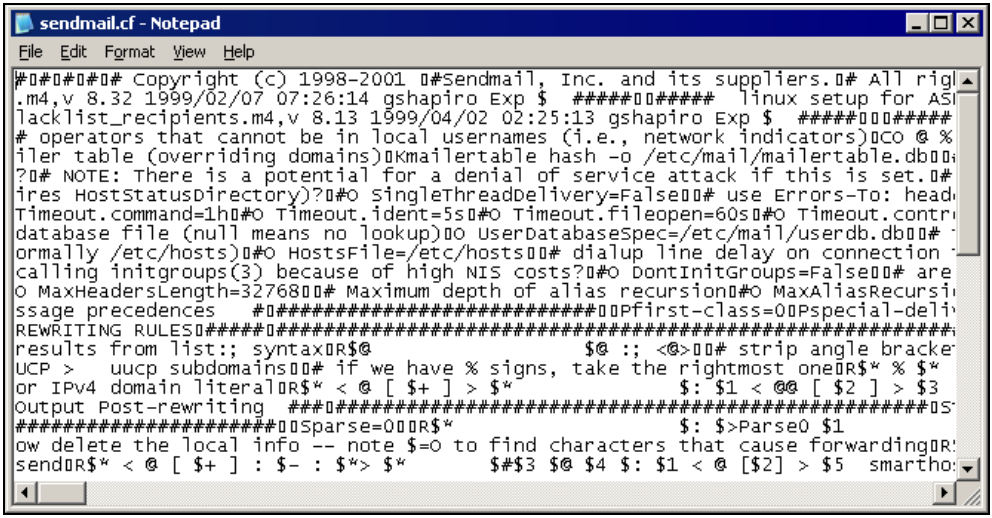


Рис. 10.2. Файл sendmail.cf, полученный с Linux-сервера в бинарном режиме

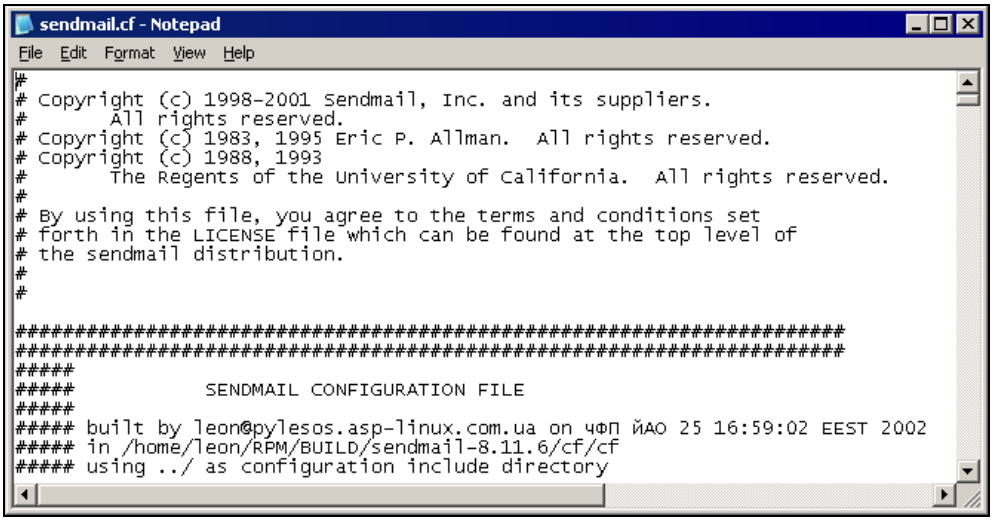


Рис. 10.3. Файл sendmail.cf, полученный с Linux-сервера в ASCII-режиме

Если передать двоичный файл из Linux в Windows в текстовом режиме, то любой символ <CR> (а он может встретиться и в таком файле) будет заменен на пару символов <CR> и <LF>, и после этого файл может стать нечитаемым.

10.1.4. Режим канала данных

Мы уже говорили о том, что для работы с FTP-сервером необходимо два канала. Порт 21 является управляющим, и по нему передаются только команды FTP. Для передачи файлов создается отдельное соединение. Этот процесс можно описать следующим образом:

1. Программа-клиент открывает порт на компьютере, куда сервер должен передать содержимое файла.
2. Серверу направляется запрос на скачивание файла и сообщается порт и IP-адрес клиентского компьютера, с которым он должен соединиться.
3. Сервер инициализирует соединение с компьютером клиента и начинает передачу данных.

Такой режим называется активным, потому что соединение устанавливает сервер. Проблема кроется в последнем действии. Если у вас установлен сетевой экран, то, скорей 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 нужно отключить сетевой экран, и он уже ничего не защитит.

Чтобы решить проблему работы через сетевой экран, был разработан пассивный режим. В большинстве серверов и клиентских программ именно его теперь начинают устанавливать по умолчанию, потому что сетевые экраны в наше время уже встроены почти во все ОС.

В пассивном режиме соединение происходит иначе:

1. Клиент запрашивает скачивание или просит принять файл.
2. Сервер осуществляет привязку к порту и сообщает клиенту номер канала, куда необходимо подключиться для получения или отправки данных.
3. Клиент устанавливает соединение с указанным портом, по которому происходит передача данных.

Таким образом, сервер только открывает порт и подготавливается к обмену файлами, а все соединения происходят только со стороны клиента. Это уже не противоречит правилам сетевого экрана.

10.2. ProFTPd

В последнее время все бóльшую популярность завоевывает сервер ProFTPd. Его мы не будем рассматривать подробно, но кратко о нем расскажем. Главный

конфигурационный файл для этого сервера — это /etc/proftpd.conf. Кроме того, к конфигурационным файлам данного сервера также относится файл ftpusers, где описываются пользователи, которым разрешена работа с FTP.

При конфигурировании proftpd большинство действий описываются блоками. Например:

```
<Limit SITE_CHMOD>
  DenyAll
  Allow from 127.0.0.1
</Limit>
```

Здесь описывается блок `Limit`, который позволяет задавать ограничение. Что именно ограничивается, указывается сразу после ключевого слова `Limit`. В данном случае это операция `SITE_CHMOD`. Внутри блока идут операторы ограничения:

- `DenyAll` — запрещает доступ со всех компьютеров;
- `Allow from` — разрешить доступ только с указанного компьютера.

Мы снова соблюдаем правило запрещения всего, что не разрешено явно. Можно также указывать следующие ограничения:

- `AllowAll` — разрешить доступ со всех компьютеров;
- `AllowGroup группы` — разрешить доступ с указанных групп;
- `AllowUser` — разрешить доступ указанным пользователям;
- `AllowOverwrite значение` — определяет, можно ли перезаписывать уже существующие файлы. В качестве значения можно указывать `on` или `off`.

Для всех пунктов, кроме последнего, есть аналогичные запреты: нужно только поменять `Allow` на `Deny`. Это операторы ограничения. В блоках могут быть и другие операторы, которые мы рассмотрим чуть позже, а сейчас давайте закончим знакомство с типами блоков.

Чтобы создать сервер с анонимным доступом, нужно создать блок `Anonymous`.

```
<Anonymous путь>
</Anonymous>
```

Анонимным пользователям будет разрешен доступ только к директории `путь` и ничего более.

Чтобы определить доступ к директории, можно использовать директиву `Directory`:

```
<Directory путь>
  DenyAll # для примера запретим доступ
</Directory>
```

В этом блоке вы можете указывать директивы, которые относятся к данной директории. Если нужно указать глобальные параметры, то используйте блок

`Global`. Глобальному блоку не нужно указывать директорию, ведь он действует на все.

Еще один блок, который может присутствовать в конфигурации сервера — `VirtualHost`. В нем указывается адрес сервера.

Теперь рассмотрим параметры, которые можно использовать внутри этих блоков:

- `DefaultRoot` — корень дерева директорий FTP, выше которого пользователь не может подняться. По умолчанию пользователь получает доступ к своему домашнему каталогу, а доступ выше запрещен;
- `DefaultTransferMode` — режим передачи данных. Их всего два — текстовый и бинарный, и в этом параметре можно указать соответственно `ascii` или `binary`;
- `DisplayConnect имяфайла` — содержимое указанного файла будет показано в приветственном сообщении во время соединения клиента с сервером;
- `DisplayLogin имяфайла` — содержимое указанного в этом параметре файла будет показано пользователю после успешной регистрации;
- `MaxClients` — максимальное количество одновременно подключающихся пользователей. Слишком маленькое число позволит хакеру с легкостью реализовать DoS-атаку, исчерпав все возможные соединения;
- `Order` — порядок, в котором сервер применяет разрешения. Если здесь указано `allow, deny`, то разрешающие записи имеют больший приоритет. Если ничего не указано, то доступ разрешается. Если же указано значение `deny, allow`, то больший приоритет имеет запрещение;
- `RootLogin` — разрешение администратору подключаться по FTP. Если здесь указано `on`, то такое действие разрешено, но лучше все его запретить, указав `off`. FTP — не лучшее решение для администрирования файлов. Если хакер каким-либо образом узнает или подберет пароль, то можете попрощаться со своими данными;
- `ServerName` — имя сервера, которое будет показываться клиенту;
- `SyslogLevel` — уровень журналирования событий:
  - `emerg` — наиболее важные;
  - `alert` — события;
  - `crit` — критические сообщения;
  - `error` — ошибки;
  - `warn` — предупреждения;

- `notice` — сообщения;
- `info` — информация;
- `debug` — отладочная информация;

- `umask маска_файла маска_каталога` — маска файла для новых файлов и папок, работающая точно так же, как параметр `umask` системы. Маска каталога не является обязательной.

Как видите, уже по параметрам понятно, что к чему, так что ничего сложного здесь нет.

10.3. Резюме

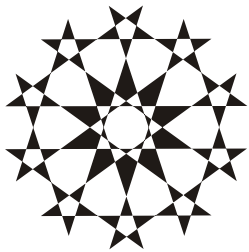
Протокол FTP и серверы различных производителей за время своего существования имели множество проблем с безопасностью. Если сложить проблемы из-за ошибок FTP и программы Sendmail, то результат может затмить даже потери от внедрения вирусов.

Главная проблема протокола FTP заключается в том, что он создавался как дружественный и удобный для пользователя. Вторая проблема — использование двух портов. Авторизация происходит только при подключении на 21 порт, а работа с каналом для передачи данных происходит без какого-либо подтверждения подлинности клиента.

Если во времена создания FTP-протокола он был действительно необходим для обмена данными, то в настоящее время от него следует избавляться. Если вы хотите дать пользователям возможность только скачивать информацию, то обратите внимание на протокол HTTP. С его помощью можно даже организовать загрузку файлов на сервер.

Если необходим обмен данными в локальной сети, то можно использовать Samba-сервер или опять же HTTP. Многие администраторы не хотят настраивать 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 протокол SFTP, который шифрует данные, поскольку такое соединение перехватить намного сложнее.



Глава 11

DNS-сервер

Каждый компьютер, подключенный к сети, должен иметь свой адрес, чтобы его можно было найти и обмениваться с ним данными. Это похоже на телефонные сети. Чтобы кому-нибудь позвонить, нужно знать 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. Отправляя письмо, вы указываете адрес, а если хотите, чтобы вам ответили, то должны указать и обратный адрес.

В эпоху зарождения сетей компьютеров в них было мало, поэтому для адресации был выбран самый простой вариант — числа. Я думаю, что если бы Интернет появлялся только сейчас, то приняли бы точно такое же решение. Но 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 было поздно, да и выбранная система IP-адресов очень удобна для обработки на программном уровне компьютерами и сетевыми устройствами. Немного поколдовав, умные люди придумали создать централизованную базу данных, устанавливающую соответствие IP-адресов и символьных имен. Таким образом, пользователь оперирует именами узлов, а программа находит нужный IP-адрес в базе DNS и уже по нему соединяется с другим компьютером или сервером.

Этот метод очень сильно упростил задачу. Раньше, чтобы найти сервер, нужно было четко знать его IP-адрес. Теперь в простой ситуации можно обойтись даже без поисковых систем типа Yahoo или Google. Например, если нужен сайт корпорации Microsoft, то можно попробовать набрать в браузере адрес **www.microsoft.com**. Таким образом, WEB-серверы фирм чаще всего можно найти просто по их имени, и не надо запоминать лишнюю информацию.

11.1. Введение в DNS

Первое время преобразование IP-адресов в имена и обратно 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 система доменных 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 могут скрываться несколько узлов. Например, хостинговые компании на одном сервере могут содержать несколько сайтов.

Сервер DNS — это большая база данных, в которой хранятся данные 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.

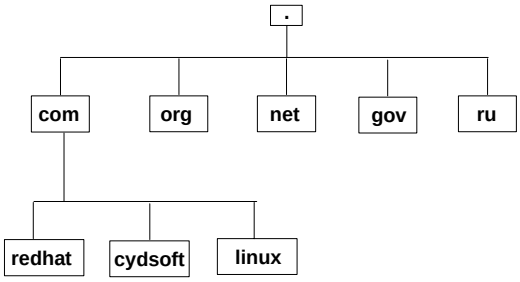


Рис. 11.1. Иерархия доменов

В качестве точки раньше выступал единственный сервер, который отвечал за корневую систему. Но однажды этот сервер оказался недоступен, и весь Интернет парализовало. С тех пор было принято решение использовать несколько серверов и распределенную систему.

Допустим, нужно найти IP-адрес сервера **www.flenov.info**. Имя разбирается справа налево. Сначала направляется запрос к корневому DNS-серверу, и он отвечает, кто обслуживает домен **info**. Затем на найденном сервере выполняется запрос на поиск домена с именем **flenov**. Если такой найден, то мы получим адрес DNS-сервера, который обслуживает **flenov.info**. И уже ему направляется запрос на получение адреса домена **www.flenov.info**. Результатом будет IP-адрес сервера, которому присвоено имя **www.flenov.info**.

Все эти действия происходят прозрачно для конечного пользователя, и когда вы набираете в браузере адрес, то никогда не увидите 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 множество 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 чаще всего есть свой DNS-сервер. Когда вы обращаетесь на какой-нибудь символьный адрес, то запрос сначала идет на сервер провайдера, затем по цепочке передается другому серверу, и так до тех пор, пока нужная запись с IP-адресом не будет найдена, если, конечно же, имя указано верно. 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: **77.15.1.190.in-addr.arpa**.

11.2. Локальный файл hosts

Мы уже знаем, что изначально для сопоставления имен и адресов использовался файл /etc/hosts. Это текстовый файл с записями типа:

```
127.0.0.1  localhost.localdomain localhost
192.168.77.1 FlenovM
```

Каждая строка — это соответствие IP-адреса его имени. По умолчанию в файле будет всего две строки, пример которых приведен выше. Первая — это петля. Напоминаю, что во всех компьютерах имя **localhost** и IP-адрес 127.0.0.1 указывают на текущую машину. Это значит, что если

нужно выполнить ping, адресовав запросы на локальный компьютер, можно написать

```
ping 127.0.0.1
```

Во второй записи устанавливается соответствие между заданным для вашего сетевого интерфейса IP-адреса и символьным именем. В данном случае моей сетевой карте присвоен адрес 192.168.77.1, и ему соответствует имя FlenovM. Это значит, что при выполнении команды `ping` можно указывать или IP-адрес, или имя компьютера. Следующие две команды идентичны:

```
ping 192.168.77.1
ping FlenovM
```

При выполнении второй команды сначала происходит обращение к файлу /etc/hosts, который вернет программе адрес 192.168.77.1, и уже на него направится эхо-запрос.

А что будет использоваться для поиска адреса первым: файл /etc/hosts или DNS-база данных? Это зависит от настроек ОС. Посмотрим на файл /etc/host.conf. В нем находится строка:

```
order hosts,bind
```

Директива `order` как раз и задает порядок просмотра. В 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 **www.redhat.com**. При этом каждый раз происходит запрос к DNS-серверу, что может приводить к задержке в пару секунд перед началом загрузки страницы. Чтобы ускорить этот процесс, можно вручную прописать в файл /etc/hosts следующую запись:

```
95.100.0.112        www.redhat.com
```

> ***Внимание!***
>
> Адрес 95.100.0.112 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, скорей 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, консоль не должна дожидаться ее завершения. Это очень удобно при запуске графических утилит, чтобы они не останавливали работу консоли. Учтите, что если закрыть окно консоли, то все программы, запущенные с ключом &, тоже закроются.

На рис. 11.2 изображено запущенное приложение для настройки DNS-сервиса. В центре показано окно, которое появляется по нажатию кнопки **Добавить**. Как видите, достаточно только выбрать тип зоны и ввести имя домена, и все готово.

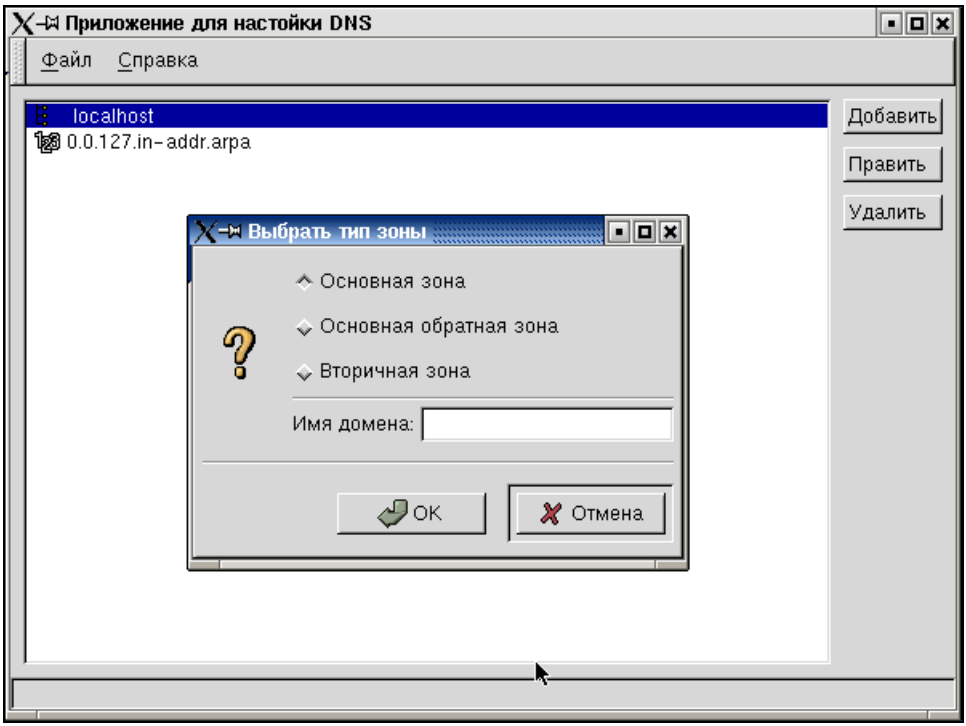


Рис. 11.2. Приложение для настройки сервиса DNS

Несмотря на наличие простой графической программы, мы рассмотрим конфигурационные файлы, которые могут использоваться сервисом DNS. Их прямое редактирование позволит сделать более тонкую настройку, и вы лучше будете понимать процесс работы DNS.

Настройка DNS-сервера начинается с файла /etc/named.conf. Пример его содержимого приведен в листинге 11.1.

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 — `zone "."`. Что это за зона в виде точки? Вспомните устройство DNS, которое мы рассматривали в начале главы. В базе данных DNS так обозначается корень. Получается, что раздел описывает корневую зону. Тип зоны `hint`, то есть хранятся лишь 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
```

А можно скачать этот файл непосредственно с FTP-сервера Internic по адресу **ftp://ftp.rs.internic.net/domain/named.root**. Этот файл нужно поместить в директорию /var/named.

Перейдем к следующему разделу `zone "sitename.com"`. Здесь описывается зона **sitename.com**. Тип записи `master`, значит ваш DNS-сервер будет главным, а все остальные 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. 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) — отображает доменное имя по его IP-адресу;
- TXT (Text, текст) — дополнительная информация, которая может содержать любое описание;
- RP (Responsible Person, ответственное лицо) — адрес E-mail ответственного за работу человека;
- HINFO (Host Information, информация о хосте) — 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). Заметьте, что @ в адресе E-mail заменено точкой, поскольку символ @ имеет особый смысл. В скобках перечислены параметры, которые для удобства расположены каждый в своей строке. Первым идет серийный номер. 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, то есть 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, в конце идут строки, определяющие преобразования. Они выглядят следующим образом:

```
имя     A       адрес
```

В нашем примере это две строки:

```
ns      A       190.12.10.1
mail    A       190.12.10.2
```

Это значит, что имени **ns.servername.com** соответствует IP-адрес 190.12.10.1, а имени **mail.servername.com** — 190.12.10.2.

11.6. Обратная зона

Теперь рассмотрим файл описания обратного преобразования IP-адреса в имя. 10.12.190.in-addr.arpa.zone 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.
```

Бóльшая часть этого файла нам уже знакома. Самое интересное хранится в последних двух строках. Здесь находится связка IP-адресов и имен серверов. Не забываем, что файл отвечает за сеть с адресами 190.12.10.*. Звездочка заменяется на число, стоящее в первой колонке, а имя, соответствующее этому адресу, указано в последнем столбце. По этому файлу мы видим следующие соотношения:

190.12.10.1 = **servername.com.**

190.12.10.2 = **mail.servername.com.**

Еще раз напоминаю, что точка в конце символьного адреса обязательна.

Для получения дополнительной информации по DNS рекомендую прочитать документы RFC 1035, RFC 1712, RFC 1706.

11.7. Безопасность DNS

Если посмотреть на задачи, которые решает служба, то кажется, что ничего 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-адреса, поэтому падение DNS для них смертельно.

Помимо вывода из строя сервера, хакер может узнать многое о структуре сети, используя DNS. Чтобы этого не произошло, желательно использовать два DNS-сервера:

- общедоступный, содержащий строки, необходимые для работы удаленных пользователей с общими ресурсами;
- локальный, видимый только пользователям вашей сети и содержащий все необходимые для их работы записи.

На локальном сервере можно так настроить сетевой экран, чтобы он воспринимал только локальный трафик и игнорировал любые попытки обращения

из всемирной сети. В этом случае злоумышленнику будет проблематично не только посмотреть базу данных DNS, но и нарушить работу сервера. Таким образом, все локальные пользователи будут лучше защищены от нарушения работы DNS и смогут спать спокойным сном под охраной сетевого экрана. Хотя спать на рабочем месте и нехорошо.

Для каждого первичного можно завести по одному вторичному серверу. Это позволит распределить нагрузку между ними и уменьшить время отклика и, конечно же, повысить отказоустойчивость. При выходе из строя одного из серверов второй возьмет на себя его функции и не позволит сети остаться без удобной возможности адресации к компьютерам по имени.

Использование парных серверов позволяет повысить производительность и безопасность. Сервисы DNS под Linux не требовательны к оборудованию. В моей сети работает четыре сервера на базе Red Hat Linux в текстовом режиме на компьютерах Pentium с частотой от 400 МГц 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 больше, чем достаточно, и хватит на ближайшие годы. Таким образом, старому компьютеру можно дать новую жизнь, причем достаточно долгую, а, главное, для компании такое решение окажется приемлемым по цене.

Однако технология создания вторичных серверов опасна. С помощью утилиты `host` хакер может добыть полную информацию о содержимом базы данных основного сервера, как это делают вторичные серверы для обновления своей базы.

Для получения базы взломщик может выполнить следующую команду:

```
host -l server.com ns1.server.com
```

В ответ на такой запрос хакер получит из базы данных все записи о сервере **server.com**. Чтобы предотвратить это, необходимо явно прописать адреса вторичных серверов в файле named.conf. Для этого в разделе `options{...}` добавляем строку:

```
allow-transfer {192.168.1.1;}
```

Такого рода ограничение можно поместить и в 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:

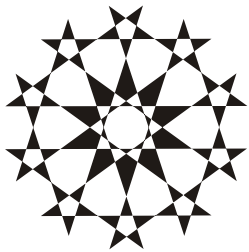
```
allow-transfer {none;}
```

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 записи DNS.

В остальном защита DNS идентична защите любого другого сервиса и ОС Linux. Как я уже говорил, лучший сервер — это тот, который выполняет узкую задачу. В нем меньше открытых портов и запущенных сервисов, поэтому его труднее взломать. Единственная сложность — большое количество серверов усложняет процесс обновления ОС. В Linux тоже бывают ошибки, и их надо исправлять. Под эту процедуру должны попасть все компьютеры, в том числе и серверы DNS.



Глава 12

Мониторинг системы

Первоначальная задача администратора — установить систему, правильно распределить права доступа и настроить все необходимые сервисы. После этого многие из них складывают ручки и начинают гонять монстров по коридорам виртуального мира Doom 3. Если вы являетесь таким администратором, то рано или поздно ваш сервер будет взломан, потому что любая система требует наблюдения. Сервер же — как ребенок: любит, чтобы за ним наблюдали, иначе нашалит.

Чтобы уменьшить вероятность проникновения в систему постороннего и 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, что 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 их 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 больше не повторится. Это ошибка, потому что хакер уже почувствовал себя безнаказанным и обязательно вернется.

К следующему его приходу вы должны быть готовы. Сделайте все возможное, чтобы найти сведения о взломщике, о том, как он проникает на сервер, и постарайтесь блокировать атаку. Также просмотрите 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, для которой он не должен быть установлен), и вы должны регулярно сканировать систему на предмет безопасности описанными ниже методами. Особенно это касается новой системы (сразу после инсталляции и настройки), но такие проверки следует производить после 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 советовал 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 остаться незалатанными на сервере. Как же поступить с ними? Неужели придется качать все сплоиты и руками проверять каждую дыру? Конечно же, нет. Существует громадное количество программ для автоматизации тестирования сервера на ошибки, и самые распространенные из них — это SATAN (сильно устаревший, но легендарный), 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 попадается 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 которых мы будем говорить чуть позже. На данный момент Internet Security Systems куплена корпорацией IBM и является ее подразделением.

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 похожи на антивирусы — защищают хорошо, но только от старых приемов. Любой новый метод взлома не будет обнаружен, пока вы не обновите программу. Поэтому я рекомендую не полагаться целиком и полностью 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 или сервис, а также 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 принимать во внимание еще и фактор конфигурации. На каждом сервере используются 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, вы пришли к другу и позвонили в дверь, но никто не открыл. Это не значит, что дома никого нет, может хозяин не услышал звонок (находился в душе, в наушниках или просто громко орал песни), или звонок не сработал. Но если позвонить по телефону, который лежит в этот момент возле хозяина, то он возьмет трубку. Возможна 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.

Существуют три метода автоматического определения уязвимости: сканирование, зондирование и имитация. В первом случае сканер собирает информацию о сервере, проверяет порты, чтобы узнать, какие установлены сервисы/демоны, и на их основе выдает отчет о потенциальных ошибках. Например, сканер может проверить сервер и увидеть на порту 21 работающую службу FTP. По строке приглашения (если она не была изменена), выдаваемой сервером при попытке подключения, можно определить его версию, которая сравнивается с базой данных. И если в базе присутствует уязвимость для 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, то есть 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 программы на наличие кода вирусов. Ситуации похожи, но искомые объекты различны. Метод хорош, но одна и та же ошибка может встречаться в нескольких программах. И если код в них разный, то сканер ее не обнаружит.

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, выявление ошибки приведет к зависанию, но вы точно будете знать о ее наличии или отсутствии.

Имитация — самый долгий, но и самый надежный способ, потому что если программе удалось взломать какой-либо сервис, то и у хакера это получится. При установке нового FTP-сервера, который еще не известен сканерам, он будет опробован на уже известные ошибки других серверов. Очень часто программисты разных фирм допускают одни и те же промахи, а 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 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 получить 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. Первым нужно автоматическое выявление ошибок для последующего применения, а вторые используют анализ, чтобы закрыть уязвимости, причем для них желательно сделать это раньше, чем найдет и использует 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, вы 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 эти биты у всех 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, значит, будут выполняться от его имени. В системе есть и 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, все-таки стоит от чего-то отказаться? Например, программа ping не является обязательной для сервера, и у нее бит SUID можно снять.

Если и после корректировки привилегированных программ осталось все равно много, то советую для начала убрать бит со всех программ. Конечно же, тогда пользователи не смогут монтировать устройства или менять себе пароль. Но нужно ли это им? 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, а иногда и невозможно, реализовать, и 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, спрятаться, и при этом иметь максимальные права. Для этого может использоваться простейший метод — установка SUID-бита на интерпретатор команд bash. В этом случае интерпретатор для любого пользователя будет выполнять команды с правами root, и взломщику для выполнения любых действий достаточно находиться в системе с гостевыми правами.

Следите за любыми появляющимися новыми программами с битами SUID или SGID и убирайте все, что вызывает подозрение.

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 так как есть определен-

ные правила конфигурирования, то есть 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 еще только наращивают свою функциональность (количество проверок).

12.3.1. lsat

Первая программа, с которой нам предстоит познакомиться — lsat. Ее история не слишком длинна, но за счет частого обновления возможности быстро выросли, а благодаря модульной архитектуре расширение функций происходит легко и быстро.

Программа lsat поставляется в исходных кодах, и найти ее можно по адресу **usat.sourceforge.net**. На момент написания этой книги была доступна версия 0.9.7. К сожалению, она не обновлялась с конца 2008 года, но, тем не менее, все равно может оказаться весьма полезной. Скачайте архив в свою домашнюю директорию, и давайте вместе установим ее и попробуем использовать. На сайте доступны tgz- и zip-версия. Я рекомендую воспользоваться первой, потому что это родной архив для Linux, и он проще в использовании.

Для установки выполните следующие команды:

```
tar xzvf lsat-0.9.7.tgz
./lsat-0.9.7.tgz/configure
./lsat-0.9.7.tgz/make
```

Первая директива распаковывает архив lsat-0.9.7.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 на выполнение используйте следующую команду:

```
./lsat-0.9.7.tgz/lsat
```

Теперь можно идти готовить кофе и медленно и печально его пить. 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 файл 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 изучить о 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:

Разработчик программы посчитал, что загрузка в графическом режиме хуже для безопасности. Действительно, это лишние программы и дополнительные проблемы. Текстовый режим не требует столько ресурсов, работает меньше программ, а значит, он быстрее и безопаснее.

Если опуститься немного 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. 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.

Далее идет список файлов, в которые могут писать пользователи каких-либо групп. Проверьте, возможно, не всем файлам из этого списка нужно давать такой статус.

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 практически в произвольном порядке, 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 это следовало бы выделять.

12.3.2. bastille

Проект bastille (**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!!!" К сожалению, и проект bastille ими заброшен: при заходе на сайт можно увидеть многообещающую надпись о том, что релиз программы будет выпущен 14 января 2008 года. Однако программа все равно остается полезной.

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 архивом RPM 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 способами, рассмотренными в самом начале книги. Хакер пытается узнать о системе все, что только можно, а администратор должен сделать так, чтобы осложнить 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 ресурсов, а иногда и немалых. Если попытаться протоколировать каждое сканирование, то злоумышленник может направить на ваш сервер пакеты, имитирующие атаки, и тогда вся вычислительная мощь сервера уйдет на выявление этих наскоков. Полу-

чится классический отказ от обслуживания (DoS), потому что сервер уже не сможет обрабатывать запросы клиентов.

Но если у вас локальный сервер в организации или просто домашний компьютер в небольшой сети, то определение любителей сканирования поможет 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 (**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, кто и когда пытался к ним обратиться. Реально эти сервисы не должны использоваться в системе. Соответствующий порт открывает программа Klaxon, и если кто-то обратился к такому порту, то это скорей 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 этот адрес и уделять ему больше внимания.

Я рекомендую установить Klaxon в системе не более чем на 2—3 сервиса, потому что слишком большое количество открытых портов вызовет подозрение у взломщика. Просто эти порты должны быть заманчивыми, то есть принадлежать к популярным сервисам. К тому же, если Klaxon возьмет на обслуживание более 5 портов, то многократное сканирование отнимет лишние ресурсы системы. Это создаст предпосылки для проведения успешной атаки отказа от обслуживания.

12.4.2. PortSentry

Эта программа была написана компанией 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 **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, если версия программы изменится к моменту выхода книги. Имя каталога будет видно в процессе разархивирования, так как программа tar 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: помимо Linux это может быть Solaris, FreeBSD, OpenBSD и т. д. При компиляции вы должны явно указать, какая ОС у вас установлена:

```
make linux
```

При установке 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 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 строки закомментированы, и вам необходимо только открыть нужные строки.

Например, вы хотите, чтобы программа следила за определенными портами. На этот случай в конфигурационном файле подготовлены закомментированные 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. 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 экран 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
```

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-адреса отправителя я подставлял чужие адреса, в результате чего сервер стал недоступным для них. Вы должны контролировать все настройки сетевого экрана, которые делает 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 программы Logcheck, описанной далее в *разд. 12.6.4*. 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.

Но перед этим желательно программу Logcheck правильно настроить. 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 адрес E-mail, на который будут отправляться 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 все настройки программы PortSentry, как показано выше, и запустил сканирование портов. CyD NET Utils (**www.cydsoft.com**) показал только первые два открытых канала. Все остальные оказались для программы закрыты, хотя реально их больше 2. 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 показанной выше строки из файла /etc/hosts.deny.

Нужно заметить, что некоторые порты могут использоваться пользователями достаточно часто, и программа будет думать, что это попытки взлома. К ним относятся порты ident (113) и NetBIOS (139), 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, порты 113 и 139 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.

При использовании LIDS 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 из того, что может делать этот пакет. Также LIDS позволяет ограничить доступ к файлам на уровне программ, а не пользователей, тем самым расширяя возможности по управлению правами доступа и увеличивается общая безопасность. Например, командам `ls`, `cat` и текстовым редакторам можно категорически запретить работу с каталогом /etc. Этим мы усложним 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 работу 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 много других любопытных вещей. Так как это один из инструментов обеспечения безопасности, мы рассмотрим журналы 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 он 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, а через 52 минуты со второго терминала вошел пользователь root.

Большинство хакеров при входе в систему выполняют эту команду, чтобы выяснить, есть ли сейчас в системе администратор. Если пользователь root присутствует, то начинающие хакеры часто уходят, так как опасаются, что их знаний не хватит, чтобы остаться незамеченными. При этом, вероятно, потом они вернутся.

Это еще одна причина, по которой администратор не должен входить в систему под учетной записью root. Лучше всего работать как простой пользователь, а когда не хватает прав, то переключаться на привилегированного. На такой случай я создал учетную запись, для которой установил UID равным нулю. Она позволяет получить доступ ко всей системе, и при этом имеет имя отличное от root, и не вызовет подозрений, когда я буду работать. Так что в моей системе 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 результата ее выполнения 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 ее выполнения приведен в листинге 12.2.

Листинг 12.2. Результат выполнения команды `lsof`

```
COMMAND PID  USER   FD  TYPE  DEVICE   SIZE     NODE NAME
init      1  root  cwd   DIR     3,2   4096        2 /
init      1  root  rtd   DIR     3,2   4096        2 /

init      1  root  txt   REG     3,2  26920   635256 /sbin/init
init      1  root  mem   REG     3,2  89547   553856 /lib/ld-2.2.5.so
init      1  root  10u  FIFO     3,2          195499 /dev/initctl
keventd   2  root  cwd   DIR     3,2   4096        2 /
keventd   2  root  rtd   DIR     3,2   4096        2 /
kapmd     3  root  10u  FIFO     3,2          195499 /dev/initctl
```

Это далеко не полный результат. Даже если 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

Следующие 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 администратора. Если взломщик пытается подобрать пароль, то вы сможете заметить быстрый рост этого файла и появление большого количество 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 макси-

мально часто, и каждой записи нужно 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 должно стать поводом для беспокойства.

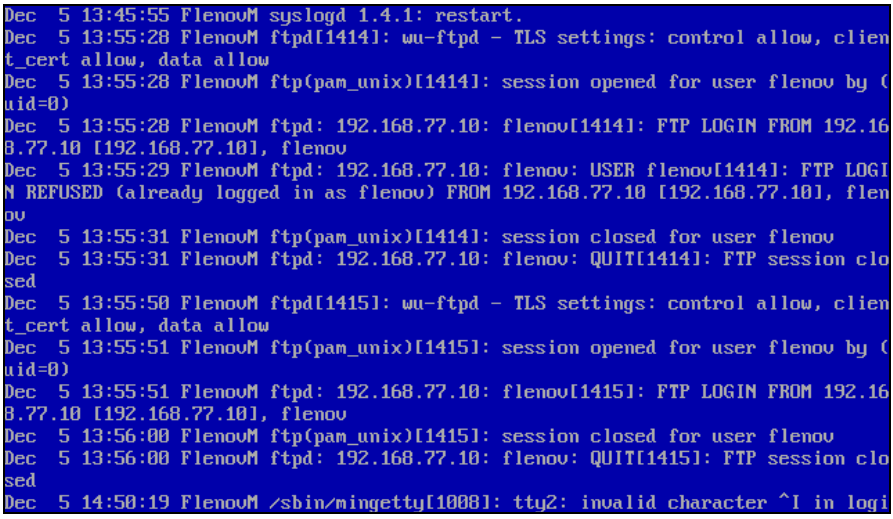


Рис. 12.1. Снимок экрана с выведенным на него файлом /var/log/messages

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.

Чтобы реагировать на подозрительные записи, вы должны быть внимательны. Самые опытные и крайне опасные хакеры используют очень интересные методы. Например, в вашей системе есть пользователь robert. Хакер видит эту запись и добавляет пользователя с именем rodert. Разница всего лишь в третьей букве, и если быть невнимательным, то ее и не заметишь, что позволит взломщику безнаказанно гулять по вашему компьютеру.

Журнал почтового сервера SendMail находится в файле /var/log/maillog. В нем можно увидеть строки примерно следующего вида (здесь три строки представляют собой одну строку журнала):

```
Jan 16 13:01:01 FlenovM sendmail[1571]: j0GA11S01571: from=root,
size=151, class=0, nrcpts=1,
msgid=<200501161001.j0GA11S01571@flenovm.ru>, relay=root@localhost
```

Из этого файла вы можете узнать, кто, когда и кому отправлял сообщения.

Вспоминается случай, когда один администратор после того, как взломали его сервер, вручную 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, все же, это не очень надежно. Умные хакеры всегда заметают свои следы и уничтожают ненужные записи из журналов. Поэтому и ручной осмотр может пригодиться, но первым делом стоит проверить 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 протокол FTP. Кто именно 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. 10.3.5*. 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 чаще всего 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 пример строки из файла /var/log/xferlog (здесь опять длинная строка разбита на две):

```
Sun Jan 16 13:21:28 2005 1 192.168.77.10 46668 /home/flenov/sendmail.cf
 b _ o r flenov ftp 0 * c
```

Из нее 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 скачан или закачан;
- 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 в случае 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 подготовленным, 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_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 минус (`-`), если оно 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 *главе 9* мы говорили о прокси-сервере squid, то упоминали и о других журналах: 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 так как 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, большинство сразу лезут 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 в них появилась информация, а потом отключить.

После этого измените директивы `ErrorLog` и `CustomLog` в файле конфигурации httpd.conf сервера Apache, указав другую директорию для хранения журналов. 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 процессорное время. Для сервера я не рекомендую этого делать. Если сейчас вы еще не прочувствовали необходимость использования журналов, то после первой проблемы или взлома системы вы осознаете все их преимущества.

Программа syslogd сохраняет в файлах журналов всю информацию о сообщениях системы. Программа klogd предназначена для сохранения сообщений ядра. Настройки журналов находятся в файле /etc/syslog.conf. Пример содержимого файла можно увидеть в листинге 12.3.

Листинг 12.3. Файл конфигурации программы syslogd

```
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
# Выводить все сообщения ядра на экран.
# Вывод других сообщений создаст на экране беспорядок.
#kern.*                                  /dev/console

# Log anything (except mail) of level info or higher.
# Don't log private authentication messages!
# Протоколировать в указанный файл все сообщения уровня info и выше.
# Исключения составляют письма, сообщения аутентификации и демона cron.
*.info;mail.none;authpriv.none;cron.none    /var/log/messages

# The authpriv file has restricted access.
# Файл authpriv содержит ограниченный доступ.
authpriv.*                                   /var/log/secure

# Log all the mail messages in one place.
# Сохранять все события почтовой системы в указанное место.
mail.*                                       /var/log/maillog



# Log cron stuff.
# Протоколировать сообщения cron.
cron.*                                       /var/log/cron

# Everybody gets emergency messages.
# Все получают критические сообщения.
*.emerg                                                    *

# Save news errors of level crit and higher in a special file.
# Сохранять сообщения новостей уровня crit (критический) и выше
# в специальный файл.
```

```
uucp,news.crit                          /var/log/spooler

# Save boot messages also to boot.log.
# Сохранять сообщения, происходящие во время загрузки в указанный файл.
local7.*                                /var/log/boot.log
```

Назначение директив легко можно проследить по комментариям. 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;
- `authpriv` — об использовании привилегированного доступа;
- `mail` — от почтовых программ;
- `cron` — от планировщиков задач `cron` и `at`;
- `deamon` — генерируемые 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` — с принтеров.

Уровень может быть одним 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`, `alert` и `emerg`.

Чем больше ошибок вы сохраняете, тем выше нагрузка на жесткий диск, да и расход ресурсов увеличивается. Для большей эффективности функционирования 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 достаточно 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 сервис cron раз в час запускает процесс, который делает резервную копию 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 затруднительным.

Для журналирования по сети в файле /etc/services должна иметься строка

```
syslog 514/udp
```

Кроме того, в файле /etc/syslog.conf должна присутствовать директива

```
сообщения     @адрес
```

Первый параметр указывает, какие сообщения нужно отправлять на сервер. Если вы хотите пересылать всю информацию, то в качестве этого параметра указывается `*.*`, если только критические, — то `*.crit`.

Второй параметр — это адрес сервера, на который будут отправляться сообщения журналов. Например, если вы хотите, чтобы все сообщения отправлялись на сервер log.myserver.com, то добавьте строку:

```
*.*        @log.myserver.com
```

Но тут есть одна проблема — для определения IP-адреса необходим DNS. При загрузке системы сервис syslogd стартует раньше DNS, поэтому определение адреса окажется невозможным. Чтобы решить эту задачу, соответствие IP-адреса и имени сервера можно прописать в файле /etc/hosts.

И последнее. На сервере служба syslogd должна быть запущена с ключом `-r`, который позволяет получать сообщения по сети и сохранять их в журнале. Для этого нужно изменить сценарий запуска сервиса. Все сценарии хранятся

в 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 параметры загрузки из этого файла, иначе они 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. В этом случае вам достаточно внести изменения в этом файле, и не надо будет трогать сценарий запуска /etc/rc.d/init.d/syslog.

Использование выделенного сервера для сохранения сообщений из журналов позволяет сохранить записи, но при этом делает их доступными для всеобщего просмотра. Дело в том, что сообщения передаются по сети в незашифрованном виде. Это не проблема, если вы отделены от Интернета сетевым экраном, и злоумышленник не смог проникнуть внутрь сети. Но если 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:

1. На сервере, отправляющем сообщения, в конфигурационном файле прописываем пересылку сообщений на локальный компьютер:

   ```
   *.*    @localhost
   ```

2. Все сообщения будут передаваться на локальный компьютер на 514 порт UDP. Чтобы все работало верно, сервер не должен работать в режиме приема сообщений, то есть запущен без ключа `-r`. Иначе порт 514 будет занят, а нам это не нужно.
3. На порту 514 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.
4. На компьютере `loagserver` создаем stunnel-сервер:

   ```
   stunnel -d 1050 -r 127.0.0.1:514
   ```

После этих действий на локальном компьютере клиент stunnel будет получать незашифрованные сообщения на порту 514, шифровать их и отправлять на порт 1050 компьютера `loagserver`. А на компьютере `loagserver` сервер stunnel будет получать зашифрованные сообщения 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 журналов 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 журнал /var/log/messages переносится 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 /etc/logrotate.conf конфигурации программы logrotate

```
# 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 следующей строки,
# чтобы файлы журналов сжимались
#compress

# RPM packages drop log rotation information into this directory
# Пакеты RPM переносят информацию о переворотах 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, 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, то есть имена журналов будут иметь вид: /etc/log/имя.X, где `x` изменяется от 1 до 4;
- `create` — требует создания нового пустого документа после смены файла журнала;
- `compress` — позволяет сжимать файлы журналов. На серверах, обрабатывающих запросы огромного количества пользователей, журналы могут занимать очень много места, и чтобы сэкономить дисковое пространство, их можно сжимать. Так как журналы содержат текст, сжатая версия может иметь объем на 70% (и даже более) меньше.

В начале файла идут описания значений по умолчанию. Затем можно указать специфичные значения для определенных журналов. В данном конфигурационном файле выделен журнала /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, так как 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, бесполезно, потому что хакер не будет добавлять записи в log-файл вручную, а воспользуется про-

стой программой на Perl или написанной прямо для 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, поскольку 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 чистить. Некоторые пользователи (да и вы сами) могут ошибиться при написании какой-либо команды и указать свой пароль. Взломщик, проанализировав файл .bash_history, увидит пароли и сможет уничтожить ваш сервер.

Если вы в командной строке набирали директиву и указывали пароль администратора root, то не поленитесь удалить соответствующую запись из пользовательского журнала. Такая ошибка может вам обеспечить бессонную ночь, а может и не одну.

Пароли администраторов могут указываться в командной строке и при использовании программы mysql. Если вы, например, выполнили команду

```
/usr/bin/mysql –uroot –ppassword
```

то она полностью сохранится в журнале. Получив доступ к вашему журналу команд bash, злоумышленник приобретает возможность использовать базу данных MySQL с правами root. В лучшем случае это будет только база данных, а в худшем (если пароль root на систему и на MySQL совпадают), — весь сервер будет под контролем взломщика.

> ***Внимание!***
>
> Никогда не выполняйте в командной строке директивы, требующие указания пароля, а если сделали это, то удалите соответствующую запись из журнала bash. В случае с MySQL нужно было задать команду `/usr/bin/mysql –uroot`. В ответ на это сервер запросит пароль, который не сохранится в журнале, а запишется только введенная команда.

Если вы работаете с сервером баз данных MySQL, то в 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. База данных — это еще не вся ОС, но и она может послужить отправной точкой для дальнейшего взлома, к тому же сами базы данных могут содержать секретную информацию, например, пароли доступа к закрытым частям 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 момент взлома вы уже пропустили.

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 вся 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 случилось. 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, он может 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, так как при этом в журнале /var/log/messages появляется запись вроде этой:

```
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 в систему, 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 своих записей, которые

будут указывать на других пользователей, тогда среди всех этих записей 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 "
```

Как обычно, обратная косая черта служит здесь для разбиения длинной команды на две строки. 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. Выделить из них реальные практически невозможно.

Хорошо, если при использовании программы `logger` хакер не будет использовать ключ `-t`, а укажет команду следующим образом:

```
logger –p kern.alert "authentication failure ; logname=robert uid=0 \
euid=0 tty=tty1 ruser= rhost= user=root "
```

В этом случае в журнале 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 программа `logger` не будет доступна хакеру, он может создать записи своей программой.

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 часто превышает 4 Мбайт, а ведь мой сайт — далеко не самый посещаемый. Это очень много текстовой информации, в которой быстро найти нужную строку практически невозможно.

Для упрощения анализа программисты и администраторы написали и продолжают разрабатывать программы-анализаторы файлов журналов. Про-

сматривать 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, и иногда необходимо видеть динамику событий. Например, одна неправильная попытка авторизации еще ни о чем не говорит, а вот 10 и более — это уже должно вызывать подозрение.

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 лично анализировать все записи и время их срабатывания. Каждый день хотя бы один пользователь может ввести ошибочный пароль, особенно если пароль, как и должен быть, сложный. Реагировать на подобные записи бессмысленно.

Есть еще один недостаток просмотра журнала построчно. Допустим, 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 директории по FTP. 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 небольшом 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 **sourceforge.net/projects/swatch**. Она 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 многих других программ. 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, так как требует наличия интерпретатора. Я уже говорил, что нельзя устанавливать на сервер то, что может открыть дверь злоумышленнику и при этом не является действительно необходимым. Без нужды я рекомендую не подключать интерпретаторы типа Perl, потому что хакеры очень часто используют их для написания собственных rootkit-программ.

12.6.3. Logsurfer

Одна из немногих программ, которая может просматривать журнал в динамике, — Logsurfer (**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.

Из-за мощных возможностей этой программы ее сложнее настраивать. Это тоже недостаток, потому что из-за ошибок в конфигурации можно не уловить очень важные события.

12.6.4. Logcheck/LogSentry

Это 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. Пароль при этом не сохраняется, чтобы хакер, прочитав журнал, не смог его увидеть. Допустим, что легальный пользователь случайно вместо имени пользователя набрал свой пароль. Такое бывает, особенно по утрам, если пользователь 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 доступен 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 bclsecurity
-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 файлами.

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`, то есть все файлы. Вторая (`xargs chmod 640`) — изменяет у всех найденных объектов права доступа на 640. Можно даже понизить это значение до 600, чтобы и читать, и писать мог только администратор.

Помимо этого, на директорию /var/log у пользователей не должно быть прав на запись, потому что это позволит злоумышленнику удалять файлы. Если хакер не сможет изменить журнал, то попытается его уничтожить. Да, вы поймете, что в системе кто-то был, но не определите, как произошло вторжение, и не сможете найти взломщика.

Помните, что, прочитав журнал, хакер может получить шанс повысить свои права, если там случайно окажется конфиденциальная информация. Но если

журналы станут доступными на запись, то взломщик сможет замести следы, удалив все записи относительно своей активности.

Но и этого недостаточно для обеспечения максимальной защиты. Если посмотреть на суть журналов, то станет очевидным, что в них ОС только добавляет новые записи. 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:

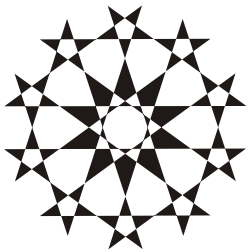
```
chattr +a /var/log/boot.log
```

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 обратно, чтобы файл снова стал доступным только для добавления записей.

Помимо самих 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.



Глава 13

Резервное копирование и восстановление

Те из вас, кто долго работает в сфере информационных технологий, уже не раз сталкивался 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 работавших в великих башнях города Нью-Йорка 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 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 данных были минимальными, нужно знать, откуда может прийти угроза. Кроме 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, а желательно и отработать процесс восстановления на практике,

используя тестовую систему, чтобы не столкнуться с неожиданностями 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, очень часто его нелепые действия приводят к уничтожению данных. При правильной политике безопасности могут быть разрушены только его собственные файлы, но и они могут иметь ценность для организации.

- Нарушение работы устройств.

  Когда я только начинал знакомиться с компьютерами, то в обиходе были дискеты 5,25 дюймов и жесткие диски максимальным размером в 20 Мбайт. Если винчестеры были относительно надежны, то информация на дискетах постоянно пропадала из-за порчи поверхности носителя. С переходом на дискеты 3,5 дюйма ситуация изменилась не сильно, а вот надежность жестких дисков со временем повышалась еще больше. Но когда мы начали оперировать гигабайтами, то в определенный момент я реально столкнулся с проблемой испорченных блоков (Bad Blocks). Примерно за полгода мне пришлось сменить три жестких диска разных производителей размером от 10 Гбайт 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 последние годы, 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 составит 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. Они запрещают выполнение любого известного злого кода. Но хакеры придумывают новые программы

и способы обхода антивирусов. И именно новые вирусы наносят максимальный ущерб, потому что для них нет еще эффективного метода лечения. Потери от эпидемий с каждым годом увеличиваются, но их можно сократить с помощью резервного копирования и восстановления.

Этот список можно продолжать еще долго, но, надеюсь, я смог вас убедить в необходимости иметь резервную копию всей значимой информации.

Посмотрим теперь, что именно нужно копировать. К важным данным можно отнести:

- системные конфигурационные файлы. На первый взгляд, это не так важно, потому что в таких файлах нет секретной и важной для организации информации. Но конфигурирование ОС и всех программ с чистого листа потребует гораздо больше времени, чем восстановление с использованием резервных файлов. А долгое время восстановления влечет за собой потери из-за простоя, 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, и компания может понести большие убытки в случае их потери;
- WEB. 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 (системы управления базами данных). И этот вопрос мы отдельно затрагивать не будем. Однако бóльшая часть теории, рассмотренной в данной главе, может в равной степени относиться как к файлам, так и к базам данных.

13.2. Доступность на все 100%

Если посмотреть на возможные причины потери данных, то видно, что наиболее вероятны 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, материнскую плату, 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. В случае сбоя основного сервера он подменяется резервным, и работу можно восстановить в максимально короткое время.

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 проста, и ее относительно недорого организовать. 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 баз данных. В случае если связь с главным компьютером прерывается, то всю работу на себя берут подчиненные серверы.

Помимо повышения надежности работы кластер может увеличивать и производительность, если все серверы функционируют параллельно и нагрузка распределена равномерно.

Еще более дешевым вариантом является использование резервных дисков. Допустим, у вас есть один сервер, который должен быть всегда доступен пользователям. В нем устанавливается дисковый массив RAID (Redundant Array of Independent Disks, избыточный массив независимых дисковых накопителей) с поддержкой зеркалирования (Mirroring), то есть RAID 1 или RAID 1+0. В этом случае RAID заботится о сохранности данных, так как их запись производится на два диска одновременно. Если один из них сломается, то полная копия сохранится на другом.

А что если выйдет из строя материнская плата или процессор? Их замена потребует времени, а мы договорились, что это недопустимо. Чтобы сократить простой в случае 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. Обеспечение сохранности данных таким способом — намного дешевле построения кластера.

В одном офисе я видел очень интересное решение. На всех клиентских компьютерах были установлены маленькие жесткие диски, на которых работала только ОС и необходимые ей файлы и программы. Помимо этого у всех 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 его на другой компьютер. Для этого у грамотного 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 резервировать данные? Однажды меня вызвали восстановить данные после выхода из строя жесткого диска. Воскресить информацию, конечно же, не удалось, потому что винчестер вышел из строя окончательно и бесповоротно, поэтому я задал вполне логичный вопрос: "А где резервная копия?" Ответ был прост (впрочем, как и владельцы компьютера), запасная 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 ломаться уже достаточно давно. Так уже 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. Помните, что никакое запасное железо не стоит так дорого, как информация и простои.

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 абсолютно 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 или ящика стола?"

Резервная копия тоже должна быть под надежной охраной. Лучше всего воспользоваться несгораемым шкафом, 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 (рис. 13.1) сделала интернет-диски удобными и доступными для пользователей Mac и Windows. На очереди и остальные системы.

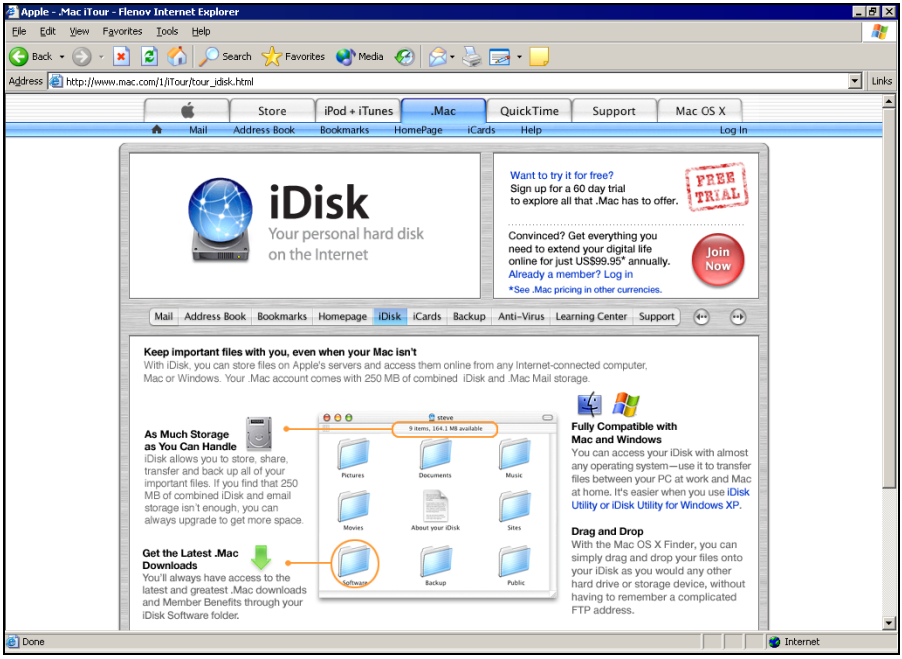


Рис. 13.1. Сайт, посвященный технологии iDisk от компании Apple

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 объема 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, как часто их использовать и как ими пользоваться? 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.

Этот список можно продолжить, но мы пока остановимся и обсудим его. Нужно четко разобраться, какие данные в системе изменяются. После этого нужно разделить 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 редко хранятся документы или необходимые для сохранения файлы. Программы из директории /bin или /usr нет смысла дублировать, потому что их легко переустановить, особенно, если сохранены их настройки.

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 иметься возможность перезаписи. Я для этих целей на данный момент использую USB-диски или флешки.

Так как конфигурация изменяется редко, то можно делать копии сразу после внесения правок. Для этого достаточно записать измененный файл на диск, и не надо копировать все конфигурационные файлы.

При восстановлении данных необходимо всегда начинать с конфигурации, в первую очередь с файлов /etc/passwd и /etc/shadow. Если этого не сделать, то ни одна программа не сможет установить правильные права доступа.

Воссоздание прав может произойти и неверно. Этому нужно уделить особое внимание, если вы применяете дополнительные средства фильтрации разрешений, используя программы, предоставляемые сторонними разработчиками.

Прежде чем сделать восстановленную систему общедоступной, необходимо убедиться, что все файлы находятся в том же состоянии, как перед сбоем. Особенно тщательно нужно проверить 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 копирования 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 производить полное копирование директории /home;
- каждый день сохранять измененные файлы.

В случае аварии восстановление должно происходить точно в той последовательности, в которой происходило резервирование: сначала восстанавливаем полную копию, а потом по очереди возвращаем на место все файлы из резервных копий. Если порядок будет нарушен, то есть риск заместить новый файл более старым.

Копирование данных по дате изменения удобно, но доступно не всегда. Большинство утилит умеют лишь обновлять существующую копию. 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 производить 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 копирования 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 независимо от файловой системы, потому что программа копирует весь диск (один к одному), используя прямой доступ к дорожкам. Восстановление полной копии гарантирует, что все права настроены четко, и программы сразу же готовы к использованию.

Но этот способ имеет достаточно много недостатков:

- требует много времени;
- производит слишком большую нагрузку на сервер;
- не реализуется средствами Linux. В большинстве дистрибутивов нет необходимых программ;
- включает в резервную копию все файлы, 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 USB-диска. Желательно иметь две одинаковых копии, потому что любые диски портятся;
- 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, по резервной копии я 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 на сервер не было 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, разумеется, не годится 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 встроено в Linux? Это простые команды копирования и архивирования, которые можно автоматизировать, прописав в планировщике задач. Если резервирование требует выполнения нескольких директив, то из них можно создать файл сценария для ОС, который будет выполнять все необходимое.

13.5.1. Копирование

Самый простой вариант создания резервной копии — использование команды `cp` (копирование файлов). Только при копировании необходимо обязательно сохранять права доступа к файлу. Вот как может выглядеть команда, сохраняющая 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, чем у документов из директории /mnt/resdisk.

13.5.2. *tar*

Копировать по одному файлу не очень удобно. Лучше, когда все находится под одной крышей. В Linux есть утилита `tar`, 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 Мбайт, 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 устройство /dev/rmt0.

Итак, для архивирования директории /home выполняем следующую команду:

```
tar cf backup.tar /home
```

При использовании параметров `cf` в архиве сохраняется и путь к файлам. 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 запустить команду разархивирования в директории /home, то папки пользователей окажутся в /home/home, а если находиться 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 файлами системы;
- `t` — просмотреть содержимое архива;
- `u` — обновить файлы в архиве, которые были изменены;
- `n date` — добавить в архив только те файлы, которые изменены позже указанной даты. Параметр `date` должен быть заменен необходимой датой;
- `P` — не удалять первый символ "/". В этом случае, в какой директории вы не запустите разархивирование, 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 перед простым копированием данных заключается в том, что результирующая копия занимает меньше места, а значит, носитель для резервирования нужен меньшего объема.

Чаще всего копируются 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. Однако 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 выйти, что затраты времени на копирование и процессорного времени на архивирование будут идентичны затратам на копирование без архивирования, или даже меньше.

Прежде чем сжимать какой-либо файл, рекомендуется подготовить tar-архив. Потом достаточно выполнить команду упаковки:

```
gzip -уровень файл.tar
```

В качестве ключа `-уровень` нужно указать степень компрессии. Максимальный уровень равен `9`. После этого указывается имя tar-архива. Давайте сожмем архивный файл, который мы создали из 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 распаковывает tar-архив.

Если необходимо из gz-файла снова получить tar-архив (без его распаковки), то можно выполнить команду:

```
gzip -d /home/backup.tar.gz
```

После этого вы снова увидите файл backup.tar, а 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 можно не использовать две команды, а 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 предназначена именно для создания резервной копии файловой системы Ext2.

Для выполнения резервной копии нужно, как минимум, указать:

- `-n` — уровень резервной копии, где `n` 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 создания 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.

Программе 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 режим командной строки, только вместо реальной файловой системы в нем вы путешествуете по архиву. При этом можно 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;
- `delete директория` — удалить из списка восстановления директорию, указанную в качестве параметра;
- `extract` — восстановить все файлы из созданного 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 ему 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:

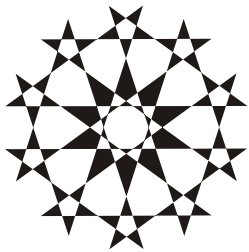
```
/usr/bin/openssl des -in /home/backup.tar.gz -out /home/backup.sec
```

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 можно разархивировать все файлы на свои места.



Глава 14

Советы хакера

В этой главе собраны некоторые короткие советы на различные темы. Создавать главу ради каждого из этих советов просто не имеет смысла, поэтому я собрал их все под одной шапкой. Надеюсь, что вам поможет эта информация.

В первом издании в этой главе был описан взлом, в текущем издании я сильно сократил эту главу и перенес часть информации на компакт-диск в документ под названием Doc/Взлом.doc.

14.1. Пароли

Не буду повторять, что пароли должны быть сложными, а обращу ваше внимание на другое. Все пароли необходимо менять через определенные периоды времени. На своем сайте (а к нему многие имеют доступ) я это делаю каждый месяц, потому что в Интернете слишком много хакеров, которые только и ищут легкую добычу.

На другие компьютеры 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 к нему не подпускаю. Не потому, что боюсь обнародовать что-то важное, а больше из-за опасения случайно потерять данные.

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. Быстрых действий можно ожидать только от того, кто проникает в систему ради уничтожения всех данных, и кому нет смысла уничтожать следы своего пребывания. Слава богу, таких взломов не так уж и много.

Итак, проникнув в систему, хакер будет незаметно сидеть в ней, и вы можете ничего не заподозрить. Но если каждый месяц меняется пароль, то после очередной смены злоумышленник теряет свои права, и 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, дает результат, если пароль несложный и постоянный. Если пароль изменяется, то вероятность успеть его подобрать до смены становится очень низкой.

Чтобы увидеть это на примере, представим, что пароль может содержать только числа. Допустим, что на первоначальном этапе он был равен 7 000 000. Хакер 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? Нет, не нужно ходить и нудить ему над ухом, есть способ проще и намного эффективнее. В Linux есть утилита chage, которая запускается следующим образом:

```
chage параметры пользователь
```

В качестве параметра можно указывать следующие ключи:

- `-m N` — минимальное число дней (`N`) до смены пароля. Указав это значение чуть меньше, чем максимальный период (см. следующий параметр), вы защитите систему от нежелательной смены паролей. Это значит, что если хакер захватит учетную запись, он не сможет изменить пароль. Конечно же, злоумышленник тоже может выполнить команду `chage`, но только если у него есть права администратора. Три-четыре дня разницы между минимальным и максимальным значением необходимы для того, чтобы пользователь смог сменить пароль, пока он не устарел. Устанавливать разницу меньше трех дней не желательно, потому что существуют выходные, и если срок действия попадет на воскресенье, пользователь не успеет сменить пароль. По умолчанию используется значение –1, что соответствует отсутствию проверки;

- `-M N` — максимальный диапазон в днях (`N`), в течение которого действует пароль. После этого пароль считается недействительным, и пользователь не сможет войти в систему. По умолчанию установлено 99999, что соответствует бесконечности, а значит, пароль никогда не устареет;
- `-d N` — дата последнего изменения пароля. Параметр `N` указывает количество дней, начиная с 1 января 1970 года. Если установить 1000, то получится 27 сентября 1972 года. Чтобы не высчитывать дату в днях, можно указать ее явным образом в формате ГГГГ-ММ-ДД;
- `-E дата` — дата окончания действия пароля;
- `-I N` — период в днях, после которого неиспользуемая учетная запись блокируется. Рекомендую указать не менее 3 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` — с этим параметром команда может вызываться любыми пользователями и позволяет им узнать информацию о времени жизни их пароля. 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 как проверить, что выбран достаточно сложный пароль, и при этом не указан снова старый? В этом нам поможет PAM-модуль pam_cracklib.so, который выполняет основные проверки и позволяет сделать их сложнее. 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 число попыток для ввода нового пароля, а они понадобятся, если пользователь попытается задать слишком простую комбинацию. Параметр `minlength` задает минимальную длину пароля.

14.2. rootkit

Проникнув в систему, хакер стремится укрепиться в ней и получить максимальные права. Например, он уже может выполнять на сервере команды от имени простого пользователя. Этого ему будет мало, поэтому следующая цель — получение прав root со всеми вытекающими отсюда последствиями.

Для решения этой задачи взломщик должен получить возможность закачивать файлы и установить в системе одну из специализированных программ, повышающих права до администратора, — такие программы называются rootkit (набор администратора). После этого взломщик выполняет команды следующим образом:

- ❒ 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 выполнять переданные ей команды с правами root? В этом помогает злополучный SGID-бит. Если он установлен, то программа будет выполняться в системе с правами администратора.

Но, к счастью, не все так просто. Для rootkit нужно еще установить SGID-бит и в качестве владельца установить пользователя root. Тут есть два пути:

- если есть возможность выполнять команды `chown` и `chmod`, то хакер сможет без проблем реализовать все необходимые действия;
- можно подменить программу на ту, которая уже имеет установленный SUID- или SGID-бит.

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. Вы должны все время следить за такими программами, и в случае появления удалять из системы. Также нужно держать под контролем все изменения, которые происходят с программами, у которых установлен бит SUID или SGID. Если их размер изменился, следует бить тревогу и восстанавливать исходное состояние программы, а также искать причину изменений.

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 со 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 цифрой 1), и, поскольку визуально программа действительно должна быть в системе, хотя и без SUID- и SGID-бита, при беглом анализе 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 для взломщика 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 **www.chkrootkit.org**. На данный момент она способна обнаружить более 60 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 и вы должны научиться самостоятельно 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 соединения 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 от имени обычного пользователя, то по умолчанию будет использоваться 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 порт окажется закрытым, то сервер должен ответить пакетом, имеющим тип RST. ОС Linux действует по стандарту, и поэтому можно легко просканировать порты с помощью этого метода. Если пакет RST не получен, то порт открыт. А вот работа Windows далека от стандарта, и здесь результат непредсказуем;

- `-sX` — TCP Xmas-сканирование. Метод похож на предыдущий, только помимо этого устанавливаются флаги `URG` и `PUSH`, указывающие на срочность данных;
- `-sN` — TCP NULL-сканирование. На сервер направляются пустые пакеты, на которые он должен сообщить об ошибке;
- `-I` — Ident-сканирование;
- `-sU` — UDP-сканирование.

Смысл сканирования в том, чтобы получить от сервера хоть какой-нибудь ответ. В зависимости от метода сканирования по положительному или отрицательному ответу определяется, закрыт порт или открыт.

Более быстрый способ получить открытые порты — это команды `lsof` (с параметром `-i`) или `netstat`, но их выполнение должно происходить локально, непосредственно с компьютера. Вторая директива будет эффективной только в том случае, если хакер в 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 нужно проверить список работающих процессов с помощью команды `ps -aux`, чтобы найти незнакомые процессы. При просмотре будьте внимательны. Вспомните пример с программой login, когда первая буква l заменялась цифрой 1. Увидев процесс 1ogin, быстрым взглядом можно ничего не заметить.

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 `пакет.rpm`.

14.3. 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 порт 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 устанавливает 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. Хакеры не любят раскрывать свои программы, потому что если они станут достоянием общественности, то лазейки, через которые взломщик проникает в систему, закроют. И все же, на закрытых сайтах можно встретить некоторые из этих разработок.

Цель этой книги — создание безопасной системы, и я не буду 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. Однако при этом надо быть внимательным и учитывать некоторые подводные камни.

При просмотре процессов нужно убедиться, что файл программы ps не модифицирован хакером. Так как исходные коды ОС Linux доступны, злоумышленник может изменить программу ps, 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,

помимо основных своих функций, программа будет играть роль потайного входа. Убедитесь, что исполняемые файлы всех работающих процессов не изменены.

К тому же, 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, так как основной процесс не модифицирован.

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.

Изменение исходных кодов — достаточно сложное занятие, и для этого нужно обладать хорошими знаниями в программировании, поэтому данный метод мало распространен, хотя он и наиболее опасен. И все же, его нельзя сбрасывать со счетов, потому что никогда не знаешь, какова квалификация проникшего в систему взломщика.

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. Поэтому обязательно проверьте все сценарии, отвечающие за загрузку сервисов, на предмет изменений. Эти сценарии находятся в директории /etc/rc.d/init.d. Сделать это может быть сложно, потому что в ОС Linux таких сценариев много. Но в любой из них хакер может добавить команды загрузки своей утилиты.

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 достаточно выполнить несколько команд, и все готово.

Как же взломщики используют ядро, чтобы спрятать свой процесс? Программа ps (и подобные ей) для определения запущенных процессов используют 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.

Это только некоторые сложности, с которыми вы можете столкнуться при поиске backdoor просмотром списка запущенных процессов, выдаваемого

программой ps. Именно поэтому нужно анализировать систему в поисках злонамеренных программ и другими способами.

Итак, помимо запуска 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 так как эта программа входит в состав Linux, то ее исходные коды также могут быть изменены. А вот от сканера портов не скроешься, правда, для его работы необходимо больше времени.

Но и от сканера портов backdoor может скрыться, точнее, он может вовсе не открывать портов. Лучший способ спрятать backdoor от сетевых анализаторов — использовать при программировании Raw Sockets (сырые сокеты), как это делают 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. Однако для прослушивания трафика сетевая карта должна работать в специализированном режиме, который легко определяется, если просмотреть состояние сетевого интерфейса командой `ifconfig`.

Есть и другой способ найти такую программу backdoor: 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.

Основной недостаток такого метода — во время работы сниффера повышается нагрузка на сервер. В этом случае все пакеты, которые проходят мимо сетевой карты, поднимаются до уровня ОС.

Но самый лучший способ защиты от backdoor — хорошо настроенный сетевой экран. Если в применяемой вами политике безопасности по умолчанию все запрещено, и разрешен только доступ к публичным ресурсам, то даже если сторонняя программа откроет какой-то порт, то подключиться к нему будет невозможно без изменения фильтров в сетевом экране. Следите за тем, чтобы никакие лишние записи в настройках Firewall не появлялись, и все мучения хакера станут напрасными.

14.4. Небезопасный NFS

Технология NFS (Network File System, сетевая файловая система) была разработана компанией Sun Microsystems в 1989 году. Идея была великолепной. Любой пользователь может монтировать каталоги сервера к своей файловой системе и использовать их, как будто они находятся на компьютере клиента. Это очень удобно в сетях. Пользовательские каталоги могут находиться на сервере и подключаться к клиенту по мере надобности. Таким образом, все файлы будут храниться централизованно, а использоваться, как будто они находятся локально.

Но, как я уже говорил, удобство и безопасность — несовместимые вещи, а NFS слишком удобна.

В состав NFS входит утилита `showmount`, которая может отобразить, какие директории и какими пользователями подключены. Для администратора это неоценимая информация.

Выполните команду `showmount -a localhost`. Вы увидите информацию о NFS на своем сервере в таком формате:

```
All mount points on localhost:
robert:/home/robert
econom:/home/jhon
buh:/home/andrey
robert:/usr/local/etc
econom:/usr/games
```

Результат разделен на две колонки символом двоеточия. В первой находится имя компьютера, подключившего удаленный раздел, а во второй — путь на сервере к подключенному ресурсу.

Подробную информацию видеть приятно, 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 пользователей системы, 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 UDP- и TCP-порту 2049 извне. Эти функции может выполнить сетевой экран. Но если хакер уже взломал какой-то компьютер в сети и получил возможность выполнять команды внутри сети, 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, то есть 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. главу 6*). Этот сервис менее болтлив и может решить ваши потребности в разделении каталогов сервера.

14.5. Определение взлома

Для эффективной защиты сервера очень важно вовремя определить, что сервер был взломан. Чем раньше вы узнаете о проникновении в систему хакера, тем скорее сможете отреагировать и предотвратить печальные последствия. Помните, взломы бывают всегда и с любой системой, но вы должны уметь их раскрывать.

Как можно выявить хакера? Существует очень много методов, и мы рассмотрим лишь наиболее интересные и эффективные.

4.5.1. 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 программирования. Лучше, если это будет C, но можно и Perl. В крайнем случае подойдет умение писать сценарии (командные файлы).

Итак, в чем заключается мой метод? Войдя в систему, хакер всегда оглядывается и старается найти способ укрепиться в системе, чтобы оставаться долгое время незаметным для администратора. Для этого взломщик чаще всего выполняет команды `who`, `su`, `cat` и др. Ваша задача установить на них ловушки. Например, можно изменить код программы `su` так, чтобы сразу после ее выполнения администратору направлялось письмо.

Получение сообщения о том, что была выполнена опасная команда, если она запускалась не администратором, — хороший повод проверить систему на наличие в ней постороннего.

Если вы не умеете программировать, можно обойтись и средствами самой ОС. Допустим, что вы хотите получать сообщения каждый раз, когда выполняется команда `who`. Взломщик часто выполняет такую директиву, когда входит в систему, чтобы 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
```

У 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 переименованный 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, которые будут вводиться 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>, чтобы выйти в командную строку, 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 почтовом ящике будет лежать новое письмо с заголовком "attack" (рис. 14.1), 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`, и 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 ничего лишнего не появлялось, и взломщик ничего не заподозрил. Хакер не должен знать, что программа отправила администратору какое-то сообщение.

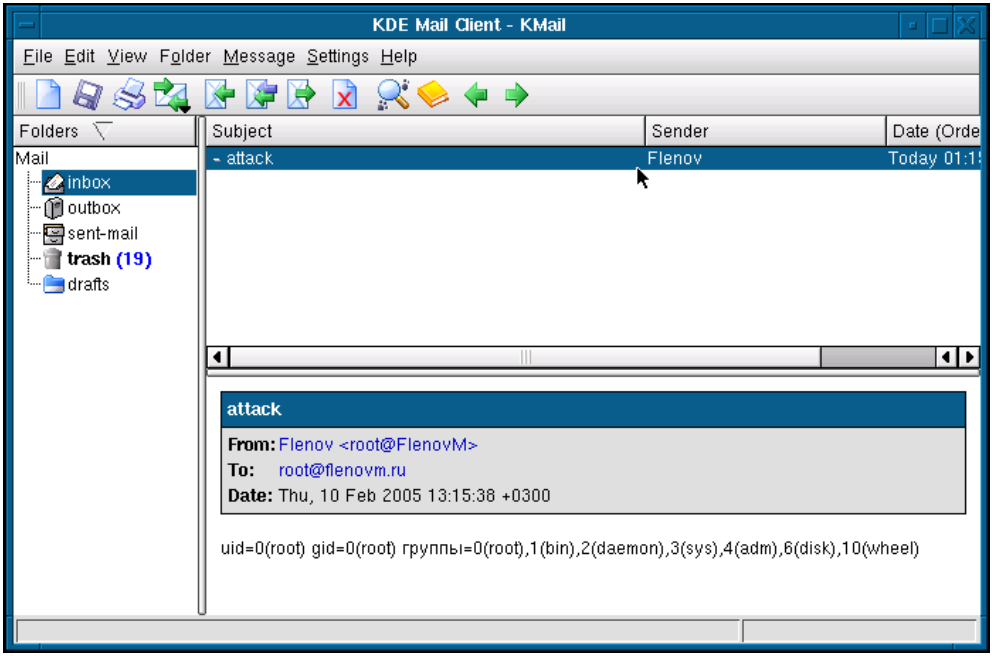


Рис. 14.1. Пример сообщения об атаке

Таким образом, можно подменить все опасные программы, которые должны быть недоступны простым пользователям.

Хакеры чаще всего не проверяют утилиты, которые запускают, а ведь угрозу можно увидеть, если выполнить команду:

```
cat /usr/bin/who
```

Вот тут и проявляется недостаток использования сценариев — их можно просмотреть. Программы, написанные на языке C и откомпилированные в исполняемый файл, при просмотре показывают абсолютно ничего не говорящий мусор.

4.5.2. Ловля на живца

Администраторы очень любят оставлять приманки для хакеров, потому что они позволяют идентифицировать злоумышленника на начальном этапе. Эта технология даже получила название honeypot (горшок меда). Как это работает? В сети устанавливается один или несколько компьютеров, в которых изначально заложены ошибки конфигурирования или легкие для подбора пароли. Основная задача этого комплекса — отслеживание обращений извне и регистрация любых взломов.

На рис. 14.2 приведена схема классической honeypot-сети. От Интернета ее отделяет сетевой экран, за которым находятся публичные ресурсы и подставные серверы/компьютеры (публичная сеть). Далее идет второй сетевой экран, который защищает и скрывает приватную сеть.

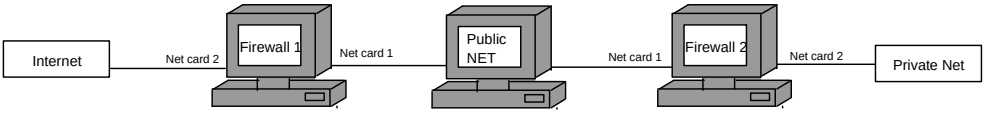


Рис. 14.2. Построение honeypot-сети

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 ложных срабатываний, его защита должна быть достаточной, дабы сервер нельзя было сломать программами, автоматизирующими поиск уязвимостей. Иначе количество ежедневно пойманных в ловушку хакеров будет исчисляться сотнями, а то и более, ведь популярные ресурсы сканируют часто.

Я настраиваю свои honeypot-серверы на максимальную безопасность, просто сетевой экран, который их защищает, 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;
- уменьшает 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 совершили ошибку в конфигурировании.

Как только замечен 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 и так 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".

14.6. Тюнинг ОС Linux

На протяжении 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 относятся к наиболее тонким, поэтому я оставил их напоследок.

Мы уже говорили о том, что лучшим средством повысить безопасность и производительность является запуск 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 минимизация возможностей. Например, сервис Apache загружает множество различных модулей, абсолютно ненужных для большинства сайтов.

Каждый лишний модуль — это очередной удар по производительности и безопасности. Поэтому их нужно отключить. Закончив с этим, переходим к более тонким настройкам.

14.6.1. Параметры ядра

Для начала откроем конфигурационный файл /etc/sysctl.conf. В нем находятся параметры ядра. Пример файла можно увидеть в листинге 14.1.

Листинг 14.1. Конфигурационный файл /etc/sysctl.conf

```
# Kernel sysctl configuration file for Red Hat Linux
# Конфигурационный файл ядра для Red Hat Linux

# For binary values, 0 is disabled, 1 is enabled.
# See sysctl(8) for more details.
# Для бинарных значений 0 — это отключен, а 1 — включен.
# Смотрите man sysctl для получения дополнительной информации

# Controls IP packet forwarding
# Контролирует переадресацию IP-пакетов
net.ipv4.ip_forward = 0

# Controls source route verification
# Контроль проверки маршрутизации от источника
net.ipv4.conf.default.rp_filter = 1

kernel.sysrq = 1

kernel.core_uses_pid = 1

#net.ipv4.tcp_ecn = 0

kernel.grsecurity.fifo_restrictions = 1
kernel.grsecurity.linking_restrictions = 1

# audit some operations
# аудит некоторых операций
kernel.grsecurity.audit_mount = 1
```

```
kernel.grsecurity.signal_logging = 1
#kernel.grsecurity.suid_logging = 1
kernel.grsecurity.timechange_logging = 1
kernel.grsecurity.forkfail_logging = 1
kernel.grsecurity.coredump = 1

# lock all security options
# блокировка всех опций безопасности
#kernel.grsecurity.grsec_lock = 1
```

Что представляют собой параметры, которые вы видите в этом файле? Попробуем разобраться на примере `net.ipv4.tcp_ecn`. На самом деле это путь к файлу относительно каталога /proc/sys, в котором все символы косой черты заменены точками. В данном случае имеется в виду файл /proc/sys/net/ipv4/tcp_ecn. 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, то есть могут быть равны 0 (отключено) или 1 (включено).

Рассмотрим параметры, которые имеет смысл изменить, а если их нет в файле, то добавить:

- `net.ipv4.icmp_echo_ignore_broadcasts` — игнорировать широковещательные эхо-пакеты протокола ICMP (параметр включен);
- `net.ipv4.icmp_echo_ignore_all` — игнорировать все эхо-пакеты протокола ICMP (значение 1). Используйте этот параметр, если не хотите связываться

с сетевым экраном. Запрет эхо-пакетов уменьшит трафик, хотя и незначительно, и при этом сделает неэффективными любые атаки с помощью ping;

- `net.ipv4.conf.*.accept_redirects` — разрешить принимать перенаправления маршрутизатора. В *главе 4* 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 * именем 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 маршрутизатора. Если в сети несколько маршрутизаторов, то параметр можно включить, чтобы распределить нагрузку между ними и не пытаться пропускать весь трафик через основной шлюз;

- `net.ipv4.conf.*.accept_source_route` — позволить принимать пакеты с маршрутизацией от источника. В *разд. 14.7.2* мы еще обсудим этот вопрос, а сейчас необходимо знать, что такие пакеты могут стать причиной обхода вашего сетевого экрана. Запретите этот параметр;

- `net.ip_always_defrag` — дефрагментировать все приходящие пакеты. Так уж повелось, что сетевой экран проверяет только первый пакет, а все остальные считает разрешенными. Хакер может обойти сетевой экран, прибегая к разбивке посылки на части (*см. разд. 14.7.1*). Если установить этот параметр, то все входящие пакеты будут дефрагментированы, и обход сетевого экрана этим методом станет невозможным;

- `net.ipv4.ipfrag_low_thresh` — определяет минимальный объем памяти, который выделяет ОС для сборки фрагментированных пакетов. Чем больше это значение, тем меньше будет манипуляций по выделению памяти. По умолчанию используется число 196 608. Слишком большое значение отнимет лишнюю память и может привести к эффекту, при котором для обработки данных не хватит ресурсов сервера. Я бы оставил значение по умолчанию;
- `net.ipv4.ipfrag_high_thresh` — определяет максимальное количество памяти (по умолчанию 262 144), выделяемое для сборки фрагментированных пакетов. Если значение превышено, то ОС начинает отбрасывать пакеты. Хакер может попытаться закидать сервер большим количеством мусорных сообщений, но сервер больше не будет реагировать на фрагментацию;
- `net.ipv4.ipfrag_time` — определяет время хранения фрагментированных пакетов в кэше. По умолчанию используется значение 30 секунд. Это очень много, за это время хакер сможет забросать весь кэш. В случае атаки на систему следует понизить это значение до 20, а то и до 10 секунд;
- `net.ipv4.tcp_syncookies` — продолжая тему защиты от DoS, я рекомендую включить этот параметр, чтобы защититься от атаки SYN Flood, при которой на сервер направляется большое количество пакетов с запросом на соединение. Хакер устанавливает в пакетах ложный обратный адрес, и сервер ожидает соединения от несуществующих или ничего не подозревающих компьютеров. Таким образом, легко превышается максимально допустимое количество подключений.

Параметров ядра очень много, и рассматривать все мы не будем. В принципе, всю необходимую информацию можно узнать из документации.

14.6.2. 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 и дальше 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
```

Из этого сообщения можно узнать 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, так как доступ можно изменить только жесткому диску в целом.

После изменения параметров их необходимо сохранить с помощью команды:

```
hdparm -k1 /dev/hda
```

После этого снова выполните команду тестирования скорости работы диска `hdparm -t /dev/hda`.

Есть еще один параметр, который влияет на производительность — режим доступа. В настоящее время 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
```

Странно, но установленные вами параметры не сохраняются после перезагрузки системы, поэтому желательно прописать эти команды в файл /etc/rc.d/rc.local. Для этого в самый конец файла добавляем три строки:

```
hdparm -m128d1c3/dev/hda
hdparm -X68/dev/hda
hdparm -k1 /dev/hda
```

14.6.3. Автомонтирование

Если вы начали знакомство с компьютером под ОС Windows, то вам покажется дикостью процесс ручного монтирования файловых систем и особенно 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 и демонтировать их очень неудобно.

Так как ОС Linux все больше поворачивается в сторону домашних пользователей, в последних дистрибутивах производителями по умолчанию включена возможность автоматического монтирования. Для этого используется служба autofs. Убедитесь, что она запускается у вас, и можно приступать к настройке.

Основной конфигурационный файл сервиса autofs — файл /etc/auto.master. Просмотрите его содержимое в листинге 14.2.

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 содержательная строка — последняя. В вашей системе она может быть закомментирована, и 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 похоже 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 используется значение 60 секунд. В большинстве случаев это вполне приемлемо.

Листинг 14.3. Содержимое файла /etc/auto.misk

```
# $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, выполнив man autofs

cd          -fstype=iso9660,ro,nosuid,nodev       :/dev/cdrom

# The following entries are samples to pique your imagination
# Следующие записи являются примерами для возбуждения 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 диска CD-ROM :

```
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 используется файловая система ISO9660 и опции, разрешающие только чтение и запрещающие использование SUID- и SGID-программ. 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. Правда, для этого нужно использовать команды Linux (например, выполнить команду `ls /misk/cd`), а не другие программы. Если просмотреть директорию /misc/cd с помощью Midnight Commander, то диск не будет смонтирован.

14.7. 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 при построении безопасного сервера или сети.

14.7.1. Дефрагментация пакетов

С помощью фрагментированных пакетов хакеры производят очень много атак на серверы. В Linux можно сделать так, чтобы ОС объединяла приходящие пакеты. Если у вас монолитное ядро (без поддержки модулей), то необходимо прописать 1 в файл /proc/sys/net/ipv4/ip_always_defrag. Это легко сделать с помощью команды:

```
echo 1 > /proc/sys/net/ipv4/ip_always_defrag
```

В последних ядрах, которые используют RPM-модули, необходимо подгрузить модуль `ip_conntrack`:

```
modprobe ip_conntrack
```

14.7.2. Маршрутизация от источника

Мы уже говорили о том, как пакеты проходят по сети. Напомню, что внутри сети пакеты передаются по MAC-адресу, а для обмена между сетями необходимо маршрутизирующее устройство, которое умеет работать с IP-адресами и направлять пакеты по нужному пути. В принципе, правильный путь определяется самим маршрутизатором. Но эти устройства управляемы, и существует несколько методов заставить их устремить 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 удобство — это "палка о двух концах". Вполне логичным было бы маршрутизацию от источника запретить, а еще 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 адрес 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 должны запретить ее на своем компьютере и тем более на компьютере, который выполняет роль шлюза в Интернет (прокси-сервер или сетевой экран). Для этого необходимо установить 0 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
```

14.7.3. 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 протокол UDP, который не поддерживает виртуальное соединение, и передача осуществляется простой отправкой пакетов в сеть без подтверждения доставки и без какой-либо авторизации, то есть 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 что шифрование используется.

14.7.4. Полный путь

Когда вы запускаете какие-либо команды или программы, то необходимо указывать полный путь к ним. Большинство пользователей и администраторов просто указывают имя запускаемого объекта, что может стать причиной взлома. Да что там говорить, я сам грешу вводом коротких команд.

Рассмотрим пример того, как злоумышленник может использовать короткие имена в своих целях на примере команды `ls`:

1. Хакер создает в каком-либо каталоге (например, в общедоступном /tmp) файл с таким же именем, как и у атакуемой программы.
2. В этот файл хакер записывет сценарий, который выполняет необходимые ему действия.
3. В переменную окружения `PATH` добавляется путь к этому каталогу.

Например, в файл может быть записан следующий код:

```
#!/bin/sh
# Изменяем права доступа к файлам /etc/passwd и /etc/shadow
chmod 777 /etc/passwd > /dev/null
chmod 777 /etc/shadow > /dev/null
# Выполняем программу ls
exec /bin/ls "$@"
```

В 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 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 каталог /tmp в самое начало системной переменной окружения `PATH`. Если теперь кто-то запустит команду `ls` без указания полного пути, то выполнится наш сценарий, который попытается изменить права доступа на файлы паролей. Если у пользователя, выполнившего команду, хватит полномочий для изменения прав, то можно считать, что система взломана.

Следите за содержимым системной переменной окружения `PATH`, чтобы ее никто не изменил.

14.7.5. Доверенные хосты

В файле .rhosts можно прописать адреса компьютеров, 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 превратится в проходной двор.

14.7.6. Защита паролей

Для защиты паролей Linux недостаточно 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 /etc/shadow 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, например, 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 для группы можно установить 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, например, 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 лишь одного есть другие запароленные службы, и 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 ввода 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.

14.7.7. 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. При этом необходимо убрать любые баннеры, которые появ-

ляются 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 внутреннего 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 порту 80) и HTTP (настроить на порт 21). Жаль, что это нельзя сделать с общедоступными сервисами, а если и можно, то бесполезно. Например, если заставить работать WEB-сервер на порту 81 вместо 80, то любой пользователь Интернета должен иметь возможность узнать об этом. А тогда и хакер будет в курсе.

14.8. 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 недоступен, 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. После этого обследуем все установленные пакеты. 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 повторного взлома. Некоторые администраторы просто возобновляют работу сервера и через некоторое 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. Чувствуя безнаказанность, хакер будет продолжать вторжения, и может возникнуть ситуация, когда взломщик успеет получить свое.

Заключение

Я надеюсь, что эта книга поможет вам больше узнать о безопасности вообще и о безопасности ОС Linux в частности. Мы достаточно много говорили и о защите, и о различных методах нападения, и может создаться мнение, что каждый администратор борется со злоумышленниками. Лично я считаю, что хакеров и взломщиков нет. Это миф, которым пытаются пугать администраторов.

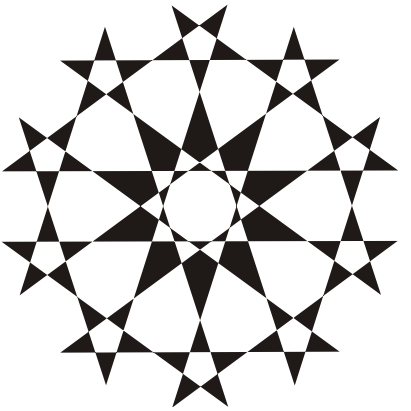
В любом мире есть сильные и слабые люди. Хакеры, в большинстве своем, — это молодые ребята, которые просто знают больше других и умеют применить свои знания на практике.

Почему-то сферу информационных технологий многие рассматривают как что-то сверхъестественное. Но это уже давно не так. Компьютеры прочно вошли в нашу жизнь и стали такими же предметами обихода, как радио или телевизор. Поэтому давайте относиться к ним также.

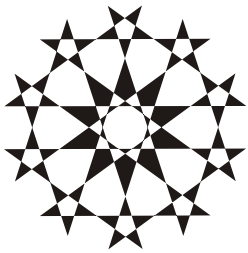
Когда автомобильный мастер вмешивается в работу двигателя, растачивает его или просто украшает машину — это нормальная ситуация. Производитель не запрещает этого делать, но и не ручается за стабильную работу после таких действий. Однако теряется только гарантия, и никому не приходит в голову преследовать автомастера как "автохакера".

Все это говорит о том, что с хакерами нужно бороться разумно. Если ваш сервер взломали, то это не значит, что надо посадить хакера. Нет, надо лишь уделять безопасности намного больше внимания. Если мы сможем повысить общий уровень знаний и качество предоставляемых в сети услуг, то взломов и хакеров будет намного меньше.

Обучайтесь, развивайтесь и улучшайте собственные, а не надейтесь на готовые решения, которые должны вас защищать. Если оставить все двери открытыми, то вас никто не защитит, в том числе и правоохранительные органы.



ПРИЛОЖЕНИЯ



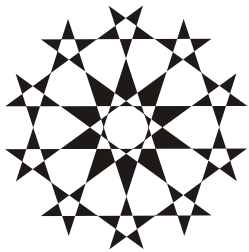
Приложение 1

Команды протокола FTP

Когда вы подключаетесь к FTP-серверу с помощью клиента, работающего из командной строки (например, sftp, упомянутого в *разд. 5.3.8*), то для работы с сервером вам помогут 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, чтобы установить права доступа 770 на файл passwd в текущей директории 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 скачанный с сервера файл;
- `put -P локальный_файл удаленный_файл` — закачать локальный файл на сервер. Выполнение команды схоже с `get`, только в данном случае локальный файл закачивается на сервер;
- `help` — отобразить список команд, которые разрешено выполнять;
- `pwd` — отобразить 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

Эти программы могут помочь вам в борьбе с хакерами. Сперва перечислим снифферы, то есть утилиты для прослушивания трафика.

- dsniff (**monkey.org/~dugsong/dsniff**) — пакет программ для прослушивания трафика, который состоит из следующих утилит:
  - dsniff — служит для перехвата паролей. Она прослушивает трафик в ожидании пакетов авторизации, и если такой пакет найден, то заветный пароль выводится на экран. Поддерживается поиск пакетов авторизации всех основных протоколов, таких как TELNET, FTP, POP и т. д.;
  - arpspoof — позволяет отправлять ARP-ответы, с помощью которых можно обмануть компьютеры жертвы, сказав, что определенный IP-принадлежит вам;
  - dnsspoof — позволяет отправлять поддельные DNS-ответы. Если жертва запрашивает IP-адрес сервера, вы можете подделать ответ DNS-сервера, чтобы вместо нужного ему сервера клиент подключился к вашему компьютеру, и вы смогли перехватить на себя трафик;
  - filesnaf — прослушивает трафик в ожидании передачи файлов по NFS;
  - mailsnaf — прослушивает сеть в ожидании E-mail-сообщений по протоколам POP и SMTP;
  - msgsnaf — отслеживает сообщения интернет-пейджеров и чатов, таких как ICQ и IRC;
  - macof — позволяет наводнить сеть пакетами с сгенерированными MAC-адресами. Если коммутатор перестает справляться с нагрузкой по определению пути, то он начинает работать как простой концентратор, и вы сможете прослушивать трафик всех компьютеров сети;

- tcpkill — может завершить чужое соединение, отправив поддельный пакет с установленным флагом RST;
- webspy — позволяет прослушивать соединения с 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 (**ettercap.sourceforge.net**) — на мой взгляд, это самая удобная программа для прослушивания трафика. Основное назначение программы — поиск паролей в пакетах всех популярных протоколов. Администраторы оценят возможность программы определять наличие в сети других программ подслушивания трафика.

Теперь напомним программы, которые помогают анализировать конфигурацию сервера и определять попытки взлома:

❐ lsat (**usat.sourceforge.net**) — программа проверки конфигурации системы (*см. разд. 12.3.1*). Анализирует конфигурацию сервера, отображая потенциальные ошибки, и в некоторых случаях может дать рекомендации по устранению недочетов;

❐ bastille (**bastille-linux.sourceforge.net**) — система выявления потенциальных ошибок в конфигурации сервера (*см. разд. 12.3.2*). Программа может автоматически исправлять ошибки и недочеты в конфигурации;

❐ Klaxon (**www.eng.auburn.edu/users/doug/second.html**) — программа для определения атак на вашу систему (*см. разд. 12.4.1*);

❐ PortSentry (**sourceforge.net/projects/sentrytools**) — утилита для слежения за портами и отслеживания попыток их сканирования (*см. разд. 12.4.2*). Позволяет автоматически сконфигурировать сетевой экран для запрета соединения с компьютером, с которого происходило сканирование;

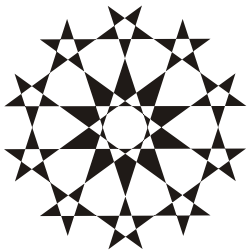
❐ Swatch (**sourceforge.net/projects/swatch**) — удобная программа анализа журналов по расписанию (*см. разд. 12.6.2*);

❐ Logsurfer (**sourceforge.net/projects/logsurfer**) — одна из немногих программ, позволяющих анализировать журнал безопасности в динамике (*см. разд. 12.6.3*).

Напоследок перечислим еще парочку полезных программ:

❐ John the Ripper (**www.openwall.com/john**) — самая знаменитая программа подбора паролей;

❐ Nmap (**nmap.org**) — сканер портов, который обладает большим количеством возможностей.

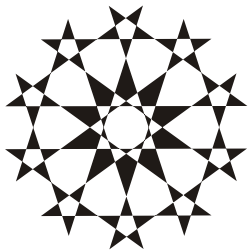


Приложение 3

Интернет-ресурсы

Я перечислю лишь несколько сайтов, которые кажутся мне наиболее информативными. Вы, вероятно, расширите этот список сайтами, интересными и полезными именно для вас.

- **www.redhat.com** — сайт компании Red Hat, где можно скачать последние версии программ, ядра, патчи и прочитать о перспективах развития ОС.
- **www.kernel.org** — сайт, посвященный ядрам ОС Linux. Здесь же можно скачать последнюю версию ядра.
- **www.securityfocus.com** — сайт посвященный безопасности, множество описаний различных уязвимостей и методов исправления ошибок.
- **www.cert.org** — еще один сайт, посвященный информационной безопасности.
- **www.securitylab.ru** — русскоязычный сайт по информационной безопасности, где можно прочитать об уязвимостях на русском языке.
- **www.xakep.ru** — сайт российского журнала "Хакер".
- **www.insecure.org** — множество полезной информации по безопасности, статьи и программы.
- **www.novell.com/de-de/linux** — сайт SUSE, одного из самых простых и удобных дистрибутивов Linux.
- **www.linspire.com** — разработчики Linspire пытаются создать ОС, в которой могли бы выполняться программы Windows на ядре Linux.
- **www.debian.org** — официальный сайт дистрибутива Debian.
- **www.slackware.com** — сайт дистрибутива SlackWare.



Приложение 4

Работа в командной строке

Небольшой секрет: когда вы набираете команды, то можно экономить время с помощью кнопки <Tab>. Нужно всего лишь начать набирать команду и нажать <Tab>. Например, если вы находитесь в корне, то там только одна директория на букву h: home. Наберите букву `h` и нажмите <Tab>, оболочка сама допишет полное имя директории. То же самое касается и файлов.

Если вам нужно повторить выполненную ранее команду, то можете использовать клавиши <↑> или <↓>. После нажатия на клавишу <↑> в командной строке появится последняя введенная команда. Нажмите еще раз, и увидите предпоследнюю. Нажмите <↓>, и вы опять вернетесь к последней команде. Таким образом можно перемещаться по истории введенных команд. Заметьте, что не обязательно вводить команду именно так, как она была введена ранее, можно ее подредактировать и лишь затем нажать <Enter>.

Псевдонимы

С помощью команды `alias` можно создавать псевдонимы для команд. Если вы часто выполняете какое-то действие, для чего нужно подать длинную команду с кучей параметров, то удобно создать для нее псевдоним. Например, для просмотра текущего каталога используется команда `ls`, но она показывает короткое содержимое, без прав доступа, а часто бывает необходимо увидеть именно их. Каждый раз указывать параметр `-I`, набирая `ls`, нудно, поэтому можно создать псевдоним `ll`:

```
alias ll="ls -I"
```

Теперь выполнение команды `ll` будет идентично выполнению `ls -I`. Но не торопитесь делать именно этот псевдоним в своей системе. Вполне возможно, что он уже у вас есть, по крайней мере, у меня в дистрибутиве такой псевдоним создан по умолчанию.

Перенаправление

Теперь поговорим о том, как можно направлять выходные данные команды не на экран, а, например, в файл. Для этого используется символ >. Например, выполните команду:

```
ls > outfile.txt
```

В результате выполнения этой команды содержимое каталога не будет выведено на экран, а будет сохранено в файл.

Если отобразить содержимое файла, подав команду

```
cat outfile.txt
```

вы увидите то, что было бы показано на экране после выполнения команды `ls` без перенаправления. Впрочем, не совсем так: в файле outfile.txt после такой команды окажется еще и его собственное имя, то есть строка outfile.txt. Если же этого файла до выполнения команды с перенаправлением не было, то и на экране бы мы его не увидели.

Надо иметь в виду, что перенаправление перезаписывает файл без подтверждения, если он уже существовал. Чтобы не переписывать файл, а дописывать в его конец, надо использовать два знака больше:

```
ls >> outfile.txt
```

Мы можем не только выводить результат в файл, но и получать параметры из файла. Например, можно выполнить команду:

```
cat < infile.txt
```

Команде `cat` в качестве параметров будет передано содержимое infile.txt.

С помощью символа вертикальной черты | можно изменять стандартный поток ввода-вывода. Допустим, вы хотите узнать, когда в систему последний раз входил пользователь по имени flenov. Для этого можно выполнить команду `lastlog`. Но если в системе зарегистрировано 1000 пользователей, то найти нужного будет проблематично. Он может находиться в любом месте, ведь результат не отсортирован. Представьте себе, если бы телефонная книга содержала всех абонентов в неотсортированном виде! Это была бы катастрофа.

Как отсортировать вывод команды? Да очень просто, нужно направить ее вывод, в данном случае список пользователей команде `sort`, которая сортирует входящие данные и выводит их в отсортированном виде. А это можно сделать с помощью символа вертикальной черты:

```
lastlog | sort
```

Запуск в фоне

Если команда работает очень долго, то вы можете запустить ее на выполнение в фоновом режиме. Для этого используется символ амперсанда (&), который нужно поставить в конце команды. Например, у меня на работе есть OLAP-сервер (это программа для получения отчетности), который написан на Java и при запуске захватывает консоль. Это значит, что консоль блокируется программой, и пока программа не будет завершена, я не могу ни закрыть консоль, ни выполнять в ней другие команды. Если таким образом запустить несколько серверов, то на рабочем столе количество консолей начинает расти. Чтобы избавиться от такого нежелательного поведения, достаточно после команды поставить символ &:

```
start-olap-server.sh &
```

Теперь программа запускается на выполнение, но не блокирует консоль. Да, вы видите вывод на экране, но в любой момент можете забрать консоль себе, нажав <Enter>. Вы можете даже закрыть консоль, и стартовавшая программа не прервет свою работу.

Последовательность команд

А что если вам нужно просто выполнить последовательность команд? В этом случае напишите эти команды через точку с запятой. Например:

```
ls > ~/outfile.txt; cat ~/outfile.txt; ls -al ~/outfile.txt
```

Эта строка идентична вводу трех отдельных команд:

```
ls > ~/outfile.txt
cat ~/outfile.txt
ls -al ~/outfile.txt
```

В первой строке я запрашиваю отображение файлов текущей директории и сохранение результата в файл outfile.txt в моей домашней директории (на домашнюю директорию указывает символ тильды (~)).

Вторая команда отображает содержимое созданного файла. Так как эта команда будет выполнена после завершения первой, файл уже будет существовать. Третья команда отображает параметры файла, его права доступа и время создания.

Если команды разделить с помощью символов &&, то вы указываете, что обе команды должны быть выполнены. Если первая команда не выполнится,

то вторая уже и не будет выполняться. Например, если файл outfile2.txt не существует, то вторая команда не будет выполнена:

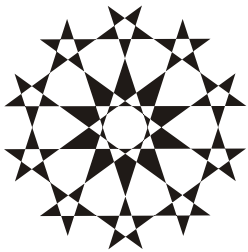
```
ls ~/outfile2.txt && cat ~/outfile.txt
```

Если же нужно выполнить одну из команд, то разделите их символами двух вертикальных черт (||). Например:

```
ls ~/outfile2.txt || cat ~/outfile.txt
```

Если файл outfile2.txt не существует, то будет выполнена вторая команда. Если же он существует, значит, первая команда будет выполнена корректно, и вторая выполняться не будет.

Командная строка в Linux — мощный механизм, который позволяет делать куда больше перечисленного. Полное описание всех возможностей нельзя поместить в таком коротком приложении, но даже упомянутые только что приемы позволят вам работать с ней более эффективно.



Приложение 5

Описание компакт-диска

На прилагаемом к книге компакте можно найти много дополнительной информации, которая позволит правильно настроить операционную систему Linux и сделать ее удобнее. Ориентироваться в ней поможет табл. П5.1.

Таблица П5.1. *Директории прилагаемого компакт-диска*

Папка	**Описание**
\Doc	Дополнительная документация по Linux
\Java	Дополнительная документация по Java
\Other	Дополнительная документация
\Source	Программы с исходными кодами
\Wallpapers	Картинки для рабочего стола

Список литературы

1. Фленов М. Программирование на C++ глазами хакера. — СПб.: БХВ-Петербург, 2004. — 336 с.
2. Фленов М. Программирование в Delphi глазами хакера. 2-е изд. — СПб.: БХВ-Петербург, 2007. — 480 с.
3. Фленов М. Компьютер глазами хакера. — СПб.: БХВ-Петербург, 2005. — 336 с.
4. **http://www.hackishcode.com/** — сайт для программистов и администраторов.
5. **http://www.flenov.info/** — сайт автора книги.
6. **http://www.xakep.ru/** — сайт журнала "Хакер".

Предметный указатель

(окончание рубрики см. на стр. 468)

Л, М