ВЗЛОМЫ и настройка LINUX

100 профессиональных советов и инструментов

Роб Фликенгер

O'REILLY®

ЭКОМ

УДК 681.3
ББК 32.97
Ф72

Фликенгер Р.
Взломы и настройка LINUX. 100 профессиональных советов и инструментов. Практ. пособ. / Фликенгер Р.; Пер. с англ.— М.: Издательство ЭКОМ, 2006.— 288 с.: илл.

ISBN 5-7163-0121-5

Книга представляет собой сборник профессиональных советов, позволяющий повысить эффективность работы серверов под управлением ОС Linux.

Рассматриваются следующие темы: основы серверов, контроль версий управляющих файлов и их резервное копирование, работа в сети, мониторинг работы сервера, вопросы защиты информации, написание сценариев на языке Perl, а также три наиболее важных программы под ОС Linux — Bind 9, MySQL и Apache.

Книга рассчитана на администраторов сетей различного уровня, а также пользователей ОС Linux, желающих глубже изучить возможности этой операционной системы.

ISBN 0-596-00461-3 (англ.)
ISBN 5-7163-0121-5 (рус.)

Как стать хакером

Слово «хакер» в современном мире имеет множество толкований. Все они изложены в так называемом Jargon File, который широко распространен в интернете. В основном, хакером называют человека, получающего радость от генерирования нестандартных, изящных решений сложных задач, имеющего очень высокую квалификацию и опыт, своего рода IT-виртуоза.

Существует целое сообщество, некая общая культура квалифицированных программистов и веб-мастеров, историю которых можно проследить до первых миникомпьютеров и эксперимента с ARPAnet — прообраза сегодняшней сети интернет. Именно члены данной культуры и использовали впервые термин «хакер». Именно они сотворили всемирную паутину. Хакеры сделали операционную систему Unix такой, какой она выглядит сегодня, запустили Usenet, заставили работать сеть интернет. Если вы являетесь частью этой культуры, внесли свой вклад в ее развитие, и другие люди, считающие себя причастными к данному сообществу, знают, кто вы такой, и называют хакером, значит, вы таковым и являетесь.

Привычный образ мыслей хакера не ограничивается только программным обеспечением. Эти люди имеют свой взгляд и на другие вещи, например, электронную технику или музыку; в принципе, вы можете встретить их среди экспертов любой области науки либо искусства. Компьютерные хакеры узнают своих собратьев везде, также называя их хакерами — и действительно, некоторые считают, что натура хакера не зависит от той среды, в которой он работает. Дальнейшее повествование будет посвящено навыкам и образу мышления именно компьютерных хакеров и традициям общей культуры, которая породила этот термин.

Существует и другая группа людей, которые во всеуслышание называют себя хакерами, но на деле таковыми не являются. Эти люди, в основном подростки мужского пола, славятся взломом компьютерных и телефонных систем. Настоящие хакеры называют таких людей взлом-

щиками (или кракерами), не желая иметь с ними ничего общего. Настоящие хакеры считают кракеров ленивыми, безответственными и не подающими особых надежд: это как умение завести машину без ключа — замкнув провода системы зажигания — не делает вас автомобильным инженером. К сожалению, многие журналисты и писатели путают эти термины, называя хакерами обычных взломщиков-кракеров, что чрезвычайно раздражает истинных хакеров. Базовое различие состоит в том, что хакеры созидают, а кракеры разрушают.

Если вы хотите стать хакером, продолжайте читать книги. А если вы мечтаете быть кракером — подключайтесь к группе новостей alt.2600 и морально готовьтесь провести следующие лет 5-10 за решеткой, после того как обнаружится, что вы не настолько умны, как себе казались. Вот и все, что я хотел сказать о кракерах.

Философия хакера

Хакеры решают проблемы и что-то создают, они верят в свободу и добровольную взаимопомощь. Чтобы быть принятым в хакерском сообществе, необходимо вести себя так, как будто полностью разделяете эту систему ценностей. А чтобы вести себя так, нужно на самом деле верить в нее и разделять.

Если же вы полагаете, что культивация такого отношения служит всего лишь способом получения одобрения в данном сообществе, то упускаете самую суть. Как в любом виде творчества, самый эффективный способ стать знатоком своего дела, мастером, заключается в умении перенять образ мышления мастера – причем не только в интеллектуальном, но и в эмоциональном плане.

Или, как говорит философия Дзен, чтобы пройти путь мастера, нужно:

«стремиться к мастеру,
следовать за мастером,
идти рядом с мастером,
видеть мастера насквозь,
стать мастером».

Поэтому, если вы хотите стать хакером, повторяйте эти строки до тех пор, пока не начнете в них верить.

1. Мир полон удивительных проблем, которые ожидают своего решения

Быть хакером интересно, однако, это именно та разновидность интересного занятия, которая требует немалых усилий, а усилия требуют мотивации. Успешные атлеты черпают мотивацию в получении физи-

ческого удовольствия от совершенствования собственного тела, заставляя себя преодолевать физические ограничения. Аналогично, вы как хакер должны получать удовольствие от решения проблем, оттачивая собственные навыки и тренируя свой интеллект.

Чтобы стать хакером, такой способ мышления необходимо сделать для себя естественным. В противном случае ваше внимание будет отвлечено такими занятиями, как секс, зарабатывание денег и повышение социального статуса.

Кроме того, необходимо развить в себе веру в способность к обучению, чтобы даже при отсутствии всей информации, необходимой для решения проблемы, взяться за решение ее части, приобрести опыт и решить, благодаря ему, следующую часть проблемы — и т.д., пока она не будет решена целиком.

2. Не нужно второй раз изобретать велосипед для решения известной проблемы

Творческий ум представляет собой ценный, но ограниченный ресурс. Не следует тратить его попусту на повторное изобретение велосипеда, когда вокруг ожидает своего часа множество новых проблем.

Чтобы научиться вести себя как хакер, нужно понимать, что время, потраченное на решение проблем другими хакерами, настолько драгоценно, что вашим моральным долгом является поделиться приобретенной информацией и найденными решениями с другими, чтобы те, другие, смогли потратить это время на решение *новых* проблем, вместо того чтобы постоянно возвращаться к старым.

Совсем необязательно делиться всеми своими творческими находками, хотя хакеры, следующие этому правилу, снискали наибольшее уважение среди других членов данного сообщества. Продажа достаточного количества своих знаний для того, чтобы иметь возможность питаться, платить за квартиру и покупать новую компьютерную технику, вполне согласуется с философией хакера. Использовать свои хакерские навыки для поддержки семьи и даже для того, чтобы сделаться богатым, пока вы помните о своей преданности делу и своих коллегах, — это правильно.

3. Скука и рутина — это зло

Хакеров, как и творческих людей в целом, никогда нельзя принуждать к скучной и монотонной работе, т.к. это будет означать, что они не используют свои способности для решения новых проблем. Подобное расточительство вредит всем. Поэтому скука и рутина не просто неприятные вещи — это истинное зло для хакера.

Чтобы научиться вести себя как хакер, нужно верить в это убеждение достаточно сильно, чтобы захотеть максимально автоматизировать все рутинные элементы работы, причем не только для себя, но и для других, особенно если они тоже являются хакерами.

Здесь существует одно очевидное исключение. Иногда хакеры делают вещи, кажущиеся скучными либо повторяющимися для стороннего наблюдателя, например, в целях «очистки мозгов» либо для приобретения навыков или практического опыта, который не может быть получен иным путем. Но все это сугубо добровольно — ни один мыслящий индивидуум не должен быть принудительно помещен в ситуацию, которая навевает на него скуку.

4. Свобода — это хорошо

Хакеры по своей природе являются противниками авторитарной власти. Любой, кто может отдавать приказы, мешает решению проблем, которые требуется решить именно сейчас. При этом, следуя образу мыслей авторитарных руководителей, будет обязательно приведена некоторая явно глупая причина, почему это нужно сделать. Поэтому хакеры обязаны бороться против всякого проявления авторитаризма, пока этот авторитаризм не успел придушить вас или других хакеров.

Не путайте борьбу с авторитаризмом с борьбой против власти любой разновидности. К примеру, детей необходимо наставлять, а преступников заключать в тюрьму, чтобы обезопасить от них общество. Хакер приемлет некоторые виды власти для получения чего-то, что, по его мнению, стоит больше, чем время, потраченное на исполнение распоряжений. Но это, как правило, сознательная сделка, действующая ограниченное время; индивидуумы, которые согласны постоянно действовать по велению авторитарных руководителей, нами не рассматриваются.

Авторитарные власти живут за счет цензуры и секретности. Они не доверяют добровольной кооперации и принципу разделения информации — им нравятся только те виды «кооперации», которые подвластны их контролю. Чтобы вести себя как хакер, нужно выработать в себе инстинктивное неприятие цензуры, секретности и применения силы либо жульничества для подчинения людей. Кроме того, нужно быть преисполненным желания действовать согласно своим убеждениям.

5. Отношение не заменяет компетенции

Чтобы стать хакером, необходимо выработать в себе положения этой системы ценностей. Но одно копирование ее не сделает вас хакером, как не сделает чемпионом по легкой атлетике либо рок-звездой. Про-

цесс превращения в хакера требует участия интеллекта, практики, верности делу и тяжелого труда.

Поэтому необходимо научиться не доверять одной только философии, но уважать компетентность любого рода. Хакеры не позволят экзаменаторам тратить свое собственное время, но они почитают компетенцию в любом виде деятельности — в особенности что касается хакерства. Компетентность в ответственных областях, с которыми немногие в состоянии справиться, ценится особенно высоко, а выше всего – в тех сферах, которые требуют точности ума, ловкости и концентрации. Если вы также чтите компетентность — вырабатывайте ее в себе. И тогда тяжкий труд и преданность делу станут признаками сильной игры, но не рутины. Такое отношение жизненно необходимо для становления в роли хакера.

Полную статью на эту тему можно найти в сети интернет по адресу: *http://www.tuxedo.org/~esr/faqs/hacker-howto.html* и в приложении книги «The Cathedral and the Bazaar» («Кафедральный собор и ярмарка»), также выпущенной издательством O'Reilly.

Эрик С. Рэймонд (Eric S. Raymond)

Эрик С. Рэймонд является автором «Нового словаря хакера», базирующегося на Файле жаргона (Jargon File) и знаменитом эссе «Кафедральный собор и ярмарка», которое послужило катализатором движения сторонников «Открытых кодов». Текст данного вступительного слова представляет собой выдержку из его статьи 1996 года «Кто такой хакер?». Рэймонд приводит аргументы в поддержку своего мнения о том, что хакерами следует называть находчивых людей, решающих интересные проблемы. Этой идеей проникнуты все книги «хакерской» серии издательства O'Reilly.

Предисловие

Хакеры способны ради любви на то,
что другие не сделают и за деньги.
—/usr/games/fortune

Слово *прием* (в данном контексте речь идет о хакерских приемах) имеет множество различных значений. «Хорошим приемом» называют наилучший способ выхода из сложившейся ситуации с использованием всех подручных ресурсов. «Дурным приемом» называют использование наименее очевидного и понятного способа решения проблемы, хотя многие «хорошие приемы» также могут показаться непонятными для несведущих.

Эффективность приема измеряется в основном способностью решения определенной технической проблемы и при этом является обратно пропорциональной количеству усилий, которые должны быть потрачены человеком, чтобы этот прием начал работать. Некоторые подходы являются масштабируемыми и даже приемлемыми. Приемы, которые используются в течение довольно долгого времени большинством хакеров, превратились в определенного рода стандарт, помогающий изобретать новые приемы. Хороший прием существует до тех пор, пока не появится лучший.

Хакерский прием раскрывает связь между абстрактным и удивительно сложным интеллектом разработчика и неоспоримым и грубым опытом человеческих потребностей. Иногда решения выглядят довольно безобразно и продолжают существовать только потому, что у кого-то «зачесались руки» их создать. Для инженера прием представляет собой полное выражение отношения «сделай сам»: никто не поймет, как усовершенствовать метод лучше, чем человек, который первый взялся за решение этой проблемы. Если личность, склонная к решению проблем, считает, что данный прием реализован безобразно, она получит отличную мотивацию для совершенствования решения — и, в конце концов, найдет его, к чему мы призываем всех читателей нашей книги.

Ведь даже самый мощный сервер с максимально возможным количеством ОЗУ и самой быстродействующей и свободно распространяемой операционной системой на планете представляет собой просто средство решения текущих проблем, используемое лишь до тех пор, пока не возникнет необходимость в лучшем, более быстром и более дешевом способе сделать это.

Какой же вывод должен быть сделан из всех этих псевдо-философских разглагольствований? Будем надеяться, что они дадут вам основу понимания того склада ума, который подсказал необходимость составления этой коллекции решений, названных нами «100 хакерских советов для сервера Linux». Одни советы являются простыми и краткими, другие достаточно сложными. Каждый из них разработан для решения конкретной технической проблемы. Надеемся хотя бы парочка советов из этой книги пригодится вам для решения наболевших проблем, будь вы новичок либо опытный администратор серверов под управлением Linux.

Структура книги

Компетентный системный администратор должен быть «мастером на все руки». Чтобы работа была максимально эффективной, необходимо уметь решать любые проблемы, которые подбрасывает система, от включения питания и до ее блокировки. Дабы помочь в этом, выносим на ваш суд эту коллекцию новаторских подходов для решения повседневных задач администратора, которые должны сэкономить ваше драгоценное время. В целях упорядочивания советов по тематике книга поделена на восемь глав:

Глава 1. Сервер: основы

Начинается с рассмотрения наиболее типичных задач, с которыми каждый день сталкиваются администраторы: управление процессом загрузки, наблюдение и регулирование использования ресурсов системой, настройка различных параметров ядра Linux ради достижения максимальной эффективности. Эту главу нельзя назвать введением в системное администрирование: скорее, ее следует считать своеобразной точкой зрения на наиболее эффективные, но не всегда очевидные технологии, которые нередко игнорируют даже опытные системные администраторы.

Глава 2. Контроль версий

Представляет собой краткий курс по использованию двух основных систем контроля версий: RCS и CVS. Возможность восстановления предыдущих версий файлов конфигурации, исходных кодов либо

документации — критичная функция, позволяющая сберечь ваш труд. Слишком многим профессиональным администраторам не хватает базовых знаний по теме контроля версий, по причине чего они предпочитают создавать неизменные резервные копии типа *.old* или *.orig*, не поддающиеся какому-либо обслуживанию. Эта глава научит работать с вышеупомянутыми системами контроля версий с помощью соответствующих команд и кратких, касающихся только данного вопроса, инструкций.

Глава 3. Резервное копирование

В этой главе рассматриваются простые и легкие в реализации методы хранения избыточных копий ваших данных. Особое внимание уделено резервному копированию по сети, утилите rsync и работе со стандартами ISO. Здесь также демонстрируется гибкость стандартных системных средств резервного копирования и даже описывается один из способов создания регулярных «мгновенных копий» состояния файловой системы (без предъявления особых требований к дисковому пространству для их хранения).

Глава 4. Сеть

Главное внимание сосредоточено не на базовой функциональности и маршрутизации, а на ряде неочевидных, но чрезвычайно полезных приемов, позволяющих использовать сеть нестандартными способами. Речь пойдет об IP-туннелях, работе с NAT и некоторых расширенных технологиях, позволяющих реализовывать интересное поведение в зависимости от задаваемых параметров. Вы когда-нибудь хотели выбирать способ обработки пакета на основе его содержимого? Прочтите данную главу.

Глава 5. Мониторинг

Представляет собой эклектичное слияние различных способов и средств, позволяющих выяснить, чем занят ваш сервер. Здесь рассмотрены некоторые стандартные и абсолютно необходимые «опциональные» пакеты, которые могут дать массу интересной информации о том, кто, где и что делает в сети. Кроме того, здесь изложено несколько способов смягчения последствий неизбежных системных сбоев, а также своевременного обнаружения попыток «нехороших» пользователей сделать с вашей сетью нечто неприемлемое.

Глава 6. Защищенная оболочка SSH

Описывает различные способы использования ssh, криптографически устойчивой и удивительно гибкой утилиты для работы в сети. Для операционной системы Linux существует несколько версий этой

утилиты, и хотя большинство рассмотренных приемов должны работать в любой версии, учтите, что их работоспособность проверялась в версии OpenSSH v3.4p1.

Глава 7. Написание сценариев

Уводит немного в сторону от основной темы книги, обращая ваше внимание на рассмотрение разнообразных наборов команд, которые просто невозможно уместить в одной командной строке. Эти методы сэкономят время и послужат в качестве примеров написания сценариев оболочки и сценариев на языке Perl.

Глава 8. Информационные серверы

Посвящена рассмотрению трех наиболее важных приложений операционной системы Linux: BIND 9, MySQL и Apache. Для прочтения главы необходимо обладать базовыми навыками по использованию этих пакетов, поскольку в ней идет разговор о способах повышения быстродействия и эффективности работы служб без особых усилий с вашей стороны. Вы узнаете, как заставить сервер работать еще быстрее, изучите методы, позволяющие легко выполнять масштабирование систем, и способы экономии времени, растрачиваемого на настройку и обслуживание.

Как пользоваться этой книгой

Возможно, вы захотите прочесть эту книгу от корки до корки, поскольку все изложенные в ней советы некоторым образом связаны друг с другом. Тем не менее, их можно рассматривать и независимо друг от друга, поскольку каждый совет представляет собой определенный способ решения конкретной задачи. С этой целью они рассортированы по темам, выделенным в отдельные главы, но при этом в тексте советов имеются перекрестные ссылки, а также ссылки на другие источники, из которых можно почерпнуть более полную информацию. Это не означает строгого разделения по темам: скорее, это служит целям систематизации схожих, но в то же время независимых методов. При чтении можно использовать и принцип веб-страниц: начать с того, что интересует в первую очередь, а обнаружив незнакомый материал, следовать по указанным ссылкам для получения более подробной информации.

Обратная связь

Вся информация, изложенная в данной книге, была тщательнейшим образом проверена, однако вполне может оказаться, что некоторые функции описанных здесь инструментальных средств были изменены

(или же мы все-таки допустили некоторые огрехи!). Просим вас сообщить нам обо всех обнаруженных в данной книге ошибках, допущенных неточностях, огрехах, путанице в выражениях либо об опечатках.

Вы можете связаться с нами по адресу:

O'Reilly & Associates, Inc.
1005 Gravenstein Hwy N.
Sebastopol, CA 95472
(800) 998-9938 (в США и Канаде)
(707) 829-0515 (для всех остальных)
(707) 829-0104 (факс)

Технические вопросы либо комментарии касательно текста книги отсылайте по электронной почте:

bookquestions@oreilly.com

Дополнительную информацию, касающуюся поддержки данной книги, включая тексты примеров и списки опечаток, вы можете найти на официальном веб-сайте по адресу:

http://www.oreilly.com/catalog/linuxsvrhack

Более подробную информацию об этой и других книгах, выпущенных издательством O'Reilly, можно найти на веб-сайте компании:

http://www.oreilly.com

Хотите поделиться своими приемами?

Вы написали собственный сценарий, которым хотели бы поделиться с другими? Посетите принадлежащий компании O'Reilly сайт, посвященный описанию различных хакерских приемов, по адресу:

http://hacks.oreilly.com

Здесь вы найдете ресурсы, связанные с темой данной книги, примеры хакерских приемов и новые хакерские методы, разработанные нашими пользователями и читателями. Кроме того, на сайте можно получить информацию о дополнительных книгах из серии «хакерские советы».

Сервер: основы

Советы № 1-22

Активная Linux-система представляет собой сложное взаимодействие аппаратного и программного обеспечения, в котором невидимые демоны выполняют распоряжения пользователя под неусыпным присмотром руководителя — ядра Linux.

Она может быть настроена на выполнение самых разнообразных задач. На персональном компьютере большая часть времени Linux тратится на управление графическим интерфейсом — прорисовкой окон и реагированием на каждое действие или команду пользователя. Такая система должна быть очень гибкой (и занимательной), в которой важнее всего быстрота реагирования и интерактивность.

С другой стороны, сервер под управлением Linux, как правило, предназначен для выполнения небольшого числа задач, связанных с перемещением информации по сети с максимальной скоростью. Если для настольной системы необходимо наличие красочных хранителей экрана и функциональности графического интерфейса пользователя, то успешным сервером Linux можно считать систему, предоставляющую доступ к информации настолько быстро и эффективно, насколько это возможно. Такой сервер должен добывать информацию из хранилища, например файловой системы, базы данных или другого места сети, и доставлять ее тем, кто в ней нуждается, будь то пользователь, подключившийся к веб-серверу или работающий с оболочкой локально либо через порт другого сервера.

В таких обстоятельствах системный администратор представляется кем-то между божеством и прислужником. В конечном счете, его работа сводится к обеспечению доступа к системным ресурсам настолько быстро и беспристрастно, насколько это вообще возможно. В его задачи входит умение проектировать новые системы, иногда основывающиеся на уже существующих решениях, а также талант и терпение прибирать за людьми, использующими систему без малейшего намека на понимание концепции «управления ресурсами».

Все эти задачи можно переложить на плечи компьютера. Пользователь считает работу системного администратора эффективной, если в его распоряжении имеются все средства, необходимые для выполнения работы. Чтобы добиться воплощения этой, звучащей неправдоподобно, задачи, системный администратор должен уметь предугадывать потребности пользователя и научиться эффективно использовать имеющиеся ресурсы.

Вначале вы познакомитесь со способами оптимизации системы под управлением Linux для выполнения только необходимых задач и предотвращения траты процессорных циклов вхолостую. Приведенные примеры покажут, как возложить на систему задачи по собственному обслуживанию и воспользоваться ее скрытыми возможностями для облегчения работы. Некоторые разделы данной главы (в частности, командная строка и управление ресурсами) описывают технологии, позволяющие воссоздать общую картину использования системы и найти способы повышения производительности.

Следующие разделы предполагают ваше знакомство с системой Linux. В частности, вы должны обладанию правами пользователя root на системе, над которой вы собираетесь экспериментировать, и комфортно чувствовать себя при работе с оболочкой командной строки. Кроме того, необходимо хорошо знать сети и стандартное сетевое оборудование. Хотя изложенные в данной книге приемы достаточно информативны, их вряд ли можно назвать введением в администрирование систем под управлением ОС Linux. Чтобы более детально разобраться в вопросах, связанных с технологиями администрирования, рекомендуем обратиться к книгам «Linux Network Administrator's Guide» и «Essential System Administration», выпущенным издательством O'Reilly & Associates.

Все советы в этой главе сгруппированы в пять блоков: время загрузки, командная строка, автоматизация, управление ресурсами, настройка ядра:

Время загрузки

- *«Отключение неиспользуемых служб»* **[Совет № 1]**;
- *«Отказ от подключения к консоли»* **[Совет № 2]**;
- *«Наиболее распространенные параметры загрузки»* **[Совет № 3]**;
- *«Создание постоянно работающего демона с помощью init»* **[Совет № 4]**.

Командная строка

- *«n>&m: перенаправление стандартных выходных данных и стандартных ошибок»* **[Совет № 5]**;
- *«Составление сложных командных строк»* **[Совет № 6]**;
- *«Работа с файлами при помощи утилиты xargs»* **[Совет № 7]**;
- *«Неизменяемые файлы в ext1/ext2»* **[Совет № 8]**;
- *«Ускорение компиляции»* **[Совет № 9]**.

Автоматизация

- *«Как чувствовать себя комфортно в оболочке командной строки»* **[Совет № 10]**;
- *«Поиск и удаление файлов с установленными битами setuid/setgid»* **[Совет № 11]**;
- *«Заставим программу sudo работать не покладая рук»* **[Совет № 12]**;
- *«Применение утилиты Makefile для автоматизации задач администратора»* **[Совет № 13]**;
- *«Подбор нового доменного имени»* **[Совет № 14]**.

Управление ресурсами

- *«Охота на пожирателей дискового пространства»* **[Совет № 15]**;
- *«Развлечения с /proc»* **[Совет № 16]**;
- *«Оперирование символьными именами процессов в procps»* **[Совет № 17]**;
- *«Распределение системных ресурсов между отдельными процессами»* **[Совет № 18]**;
- *«Удаление "хвостов" за бывшими пользователями»* **[Совет № 19]**.

Настройка ядра

- *«Удаление неиспользуемых драйверов из ядра системы»* **[Совет № 20]**;
- *«Использование больших объемов ОЗУ»* **[Совет № 21]**;
- *«hdparm: тонкая настройка параметров IDE-устройств»* **[Совет № 22]**.

СОВЕТ № 1 Отключение неиспользуемых служб

(Настройте сервер для использования только необходимых служб)

Установка сервера подразумевает создание системы, выполняющей определенные функции с максимально возможной скоростью и эффективностью. Хотя внешние службы не имеют никакого отношения к внутренним, они могут потреблять ресурсы и в некоторых случаях создавать полнейшую неразбериху в работе служб, абсолютно не связанную с основными обязанностями сервера.

Устанавливая сервер, необходимо постоянно спрашивать себя: что должна делать данная система? Действительно ли необходимо наличие службы FTP на веб-сервере? Нужна ли служба NFS (сетевой файловой системы) на DNS-сервере, если не экспортируются никакие разделяемые ресурсы? Нужна ли служба автоматического подключения томов (automount), если все тома устанавливаются статически?

Чтобы выяснить, какие функции выполняет ваш сервер, просто запустите команду ps ax. Если в данный момент к системе никто не подключен, то результатом выполнения команды будет список активных процессов в системе в текущий момент времени. Можно просмотреть

список программ, для которых демон inetd разрешает установку подключений, с помощью команды grep -v ^# /etc/inetd.conf либо (что ближе к теме) команды netstat -lp. Первая команда выведет на экран все незакомментированные строки файла *inetd.conf*, тогда как вторая команда, запущенная с правами пользователя root, выведет информацию по всем портам, находящимся в режиме прослушивания. В идеале можно свести объем информации, выводимой командой ps ax, к одной странице и менее, запретив серверы с ветвлениями, такие как httpd.

Ниже приводится список общеизвестных служб, включаемых по умолчанию во многие дистрибутивы:

portmap, rpc.mountd, rpc.nfsd

Службы являются частью сетевой файловой системы (NFS). Запущен ли на вашей системе сервер NFS? Требуется ли подключение удаленных разделяемых ресурсов NFS? Если нет ответа «да» хотя бы на один из этих вопросов, то вышеупомянутые службы не нужны. Освободите занимаемые ими ресурсы и избавьте себя от потенциальных угроз вашей безопасности.

smbd и *nmbd*

Эти демоны имеют отношение к серверу Samba. Нужно ли экспортировать разделяемые ресурсы блока серверных сообщений (SMB) в Windows-системы или на другие компьютеры? Если «нет», тогда можно отключить эти процессы.

automount

Программа для автоматической установки файловых систем предоставляет доступ по требованию к локальным либо сетевым файловым системам, не требуя прав пользователя root. Функция удобна на клиентских системах, в которых пользователю необходим доступ к сменным носителям информации (CD или дискеты) либо сетевым ресурсам. В то же время на сервере, выделенном для выполнения конкретных задач, данная служба абсолютно не нужна. Если система не предоставляет консольный доступ и не использует сетевые разделяемые ресурсы, отключите службу automount и задайте установку всех необходимых файловых систем статически, в файле */etc/fstab*.

named

Для выполнения разрешения сетевых имен предназначены файл настроек */etc/resolv.conf* и библиотеки привязки. Если вам не требуется служба доменных имен для обслуживания других компьютеров и не запущен сервер кэширования DNS (см. *«Настройка кэширования DNS с правами для локальных доменов»* **[Совет № 78]**), служба *named* не нужна.

lpd

Печатаете ли вы когда-нибудь с помощью данного компьютера? Конечно, такая вероятность существует, если с помощью этого сервера осуществляется навигация по сети интернет, однако в любом случае подобная система не должна принимать запросы на печать. Удалите демон печати, если не планируется его использовать.

finetd

Если используется служба ssh, запущенная в автономном режиме, либо вы работаете только с автономными программами (такими как Apache, BIND, MySQL либо ProFTPD) — наличие *inetd* излишне. Проверьте, запуск каких служб задан при помощи команды grep –v –etc/inetd.conf. Если все службы закомментированы, тогда зачем вообще запускать демона? Удалите его из перечня автозагрузки: из системных ресурсов (rc) либо с помощью простой команды chmod –x /usr/sbin/inetd).

telnet, rlogin, rexec, ftp

Функции этих служб, связанные с удаленным подключением к системам, удаленным выполнением программного кода и передачей файлов, сегодня взяли на себя их аналоги: службы ssh и scp, обладающие более гибкими возможностями и защитой с помощью шифрования. Если нет достаточно веских оснований, чтобы оставить вышеупомянутые службы, желательно удалить их из системы. При необходимости поддержки подключений ftp воспользуйтесь дополнительным модулем mod_sql для proftpd (см. раздел *«Использование proftpd с источником аутентификации mysql»* **[Совет № 85]**).

finger, comsat, chargen, echo, identd

Применение служб finger и comsat являлось оправданным во времена открытого интернета, когда пользователи были чрезвычайно любопытными, но при этом, в основном, имели хорошие намерения. В наши дни наличия программ для скрытого сканирования портов и удаленного использования ошибок переполнения буфера, запуск внешних служб, предоставляющих информацию о сервере, принято считать *плохой* идеей. Порты, используемые службами chargen и echo, применяемые когда-то для тестирования работоспособности сети, теперь являются слишком привлекательными для несанкционированного использования сервера.

И, наконец, служба identd, которая когда-то являлась важным и значимым источником информации о подключенных к удаленным серверам пользователях. К несчастью, в наши дни программного кода для эксплуатации привилегий пользователя root и распространения

настольных систем под управлением Linux установленная служба identd, обычно дающая ложную информацию о подключенных к системе пользователях, встречается так часто, что большинство вебсайтов просто игнорируют эту информацию. А поскольку служба identd перестала быть заслуживающим доверия источником информации, зачем вообще ее включать?

Для отключения неиспользуемых служб необходимо вначале завершить их работу: при помощи команды service stop в папке */etc/rc.d/init.d* путем удаления их из файла */etc/inetd.conf* либо уничтожая их вручную командой kill. Удалите записи о службах из файлов */etc/rc.d/**, чтобы они не запускались во время следующей загрузки. Избавившись от всех ненужных служб, перезагрузите систему и проверьте таблицу активных процессов еще раз.

Если нельзя обойтись без запуска на вашей системе незащищенных служб, воспользуйтесь упаковщиками tcp либо локальным брандмауэром для предоставления доступа только тем клиентам, которые действительно в нем нуждаются.

Дополнительная информация:

- *«Создание брандмауэра из командной строки любого сервера»* **[Совет № 45]**;
- *«Установка кэширования DNS с правами локальных доменов»* **[Совет № 78]**;
- *«Использование proftpd с источником аутентификации mysql»* **[Совет № 85]**.

СОВЕТ **№ 2**

Отказ от подключения к консоли

(Доступ к компьютеру без пароля)

Однажды это обязательно случится — вам понадобится поработать на компьютере вместо друга или клиента, сменившего пароль пользователя root, на котором у вас отсутствует собственная учетная запись.

Если имеется доступ к консоли и компьютер не перезагружался, лучшим решением является загрузка системы в однопользовательском режиме. Обычно после нажатия клавиш Ctrl-Alt-Del необходимо дождаться окончания тестирования POST, чтобы передать параметр single в ядро загрузки. Например, в приглашении командной строки LILO это будет выглядеть как:

```
LILO: linux single
```

Результат — благополучное попадание в оболочку пользователя root. Однако на некоторых системах, например RedHat, возможно получение следующего предупреждения:

```
Give root password for maintenance
or type Control-D for normal startup
/* Задайте пароль пользователя root для выполнения обслуживания
/* либо нажмите Ctrl-D для продолжения обычной загрузки
```

Если бы пароль пользователя root был известен, проблемы не было бы. При условии определенного везения, сценарий инициализации (init) разрешит вам нажать ^C на данном этапе и попасть в приглашение командной строки для пользователя root. Однако подавляющее большинство процессов init «умнее», и в них осуществляется перехват нажатия этой комбинации. Как действовать в таком случае? Разумеется, всегда можно загрузиться с диска аварийного восстановления и сбросить пароль, однако, предположим, что его нет под рукой либо в компьютере отсутствует CD-привод. Даже в этом случае еще не все потеряно! Наберите в приглашении командной строки LILO:

```
LILO: linux init=/bin/bash
```

Эта команда вместо запуска процесса */sbin/init* и перехода к выполнению обычных процедур из */etc/rc.d/** заставляет ядро ограничиться предоставлением доступа к оболочке командной строки. Никаких паролей, никаких проверок файловой системы, и в то же время быстрое получение доступа к приглашению командной строки с полномочиями пользователя root.

К сожалению, файловая система пользователя root будет доступна только для чтения (поэтому нельзя ни проверить ее, ни переустановить в качестве доступной для чтения/записи). Кроме того, нельзя работать с сетью, а в системе не будет запущен ни один из обычных системных модулей. Все что можно сделать с этим приглашением командной строки – это сбросить пароль либо настроить один-два файла. Но главное: не нажимайте ^D и не набирайте Exit, поскольку в этом случае оболочка командной строки и системное ядро решат, что вы хотите вообще выйти из операционной системы Linux. Как же работать с файловой системой в ситуации, когда она является доступной только для чтения? Попробуйте следующую команду:

```
# mount -o remount,rw /
```

Результатом ее выполнения станет установка файловой системы root в качестве доступной для чтения и записи. Далее воспользуйтесь командой passwd для изменения пароля пользователя root и, если преды-

дущий администратор утерял пароль, подумайте, как сделать так, чтобы он не потерял его в дальнейшем.

После сброса пароля НЕ ПЕРЕЗАГРУЖАЙТЕ КОМПЬЮТЕР. Поскольку в это время никакой сценарий init не запущен, безопасно завершить работу системы не удастся. Самый быстрый способ безопасного завершения работы компьютера заключается в повторной установке файловой системы:

```
# mount -o remount,ro /
```

Установив файловую систему с доступом только для чтения, нажмите на кнопку Reset, перейдите в однопользовательский режим и приступите к реальной работе.

СОВЕТ № 3

Наиболее распространенные параметры загрузки

(Изменение параметров ядра во время загрузки)

Как показано в разделе *«Отказ от подключения к консоли»* **[Совет № 2]**, задание параметров ядра в приглашении командной строки LILO позволяет менять программу, запускаемую в ходе загрузки системы в первую очередь. Изменение сценария init с помощью команды init=/bin/bash — это лишь одна из множества полезных опций, которая может задаваться во время загрузки. Наиболее часто используемые параметры загрузки таковы:

single

Позволяет выполнять загрузку в однопользовательском режиме.

root=

Изменяет устройство, установленное в качестве /. Например, команда:

```
root=/dev/sdc4
```

включает загрузку с четвертого раздела третьего диска scsi (вместо загрузочного раздела, заданного по умолчанию).

hdX=

Позволяет менять геометрию IDE-устройства. Эта функция полезна в случае выдачи BIOS некорректной информации:

```
hda=3694,255,63 hdd=cdrom
```

Команда задает в качестве ведущего IDE-устройства жесткий диск объемом 30 Гб в режиме LBA, а в качестве ведомого IDE-устройства — CD-привод.

console=

Задает консоль последовательного порта для ядра, поддерживающего данную функцию:

```
console=ttyS0,19200n81
```

С помощью этой команды задается вывод сообщений загрузки на устройство ttyS0 (первый последовательный порт), на скорости 19200 бод, без контроля четности, 8 битов данных, 1 стоп-бит. Обратите внимание: чтобы получить доступ к настоящей последовательной консоли, необходимо добавить в файл */etc/inittab* следующую строку:

```
S1:12345:respawn:/sbin/agetty 19200 ttySo vt100
```

nosmp

Отключает в ядре функцию SMP, если она включена. Это позволит решить возможные проблемы на мультипроцессорной системе.

mem=

Определяет общее количество свободной памяти (см. раздел *«Использование больших объемов ОЗУ»* **[Совет № 21]**).

ro

Устанавливает раздел / в качестве доступного только для чтения (обычно так и происходит, а затем, после запуска fsck, он устанавливается повторно с правами доступа на чтение/запись).

rw

Задает доступ к разделу / на чтение/запись. Задайте в команде init параметры rw

```
init=/bin/bash rw,
```

чтобы впоследствии не пришлось выполнять команду mount -o remount,rw, упомянутую в разделе *«Отказ от подключения к консоли»* **[Совет № 2]**.

Кроме того, можно передавать параметры для контроллеров SCSI, устройств IDE, звуковых карт, т.е. практически любого аппаратного обеспечения. Каждый драйвер имеет свои отличия, но обычно они позволяют задавать прерывания, базовые адреса ввода-вывода, контроль четности, скорость, параметры автоматического поиска устройств (auto probing) и т.д. Более подробно об этом можно прочесть в онлайновой документации.

Дополнительная информация:

- руководство пользователя по параметрам загрузки (команда man bootparam);
- */usr/src/linux/Documentation/**.

СОВЕТ №4

Создание постоянно работающего демона с помощью init

(Убедитесь, что процесс работает независимо от условий)

Существует масса способов автоматического перезапуска процесса в случае непредусмотренного завершения его работы. Простейший пример такого сценария приведен ниже:

```
$ while : ; do echo "Run some code here..."; sleep 1; done
```

Если вместо строки *echo* запустить фоновый процесс, то он будет выполняться непрерывно или, по крайней мере, будут постоянно производиться попытки его запуска. Символ : заставит цикл while работать бесконечно (такой прием более эффективен, чем /bin/true, поскольку в этом случае не потребуется создание внешней команды для каждой итерации). Поэтому *не* запускайте фоновый процесс вместо команды echo, если не хотите загрузить таблицу процессов — ведь команда while запустит столько фоновых процессов, сколько сможет, делая это каждую секунду. Однако если мы хотим реализовать более серьезный хакерский прием, то функциональности команды while явно не хватит.

Что происходит, когда процесс попадает в аварийные условия? Он немедленно завершается, после чего система будет пытаться запустить его каждую секунду, не сообщая о возникновении проблемы, если только в процессе не предусмотрено ведение собственного журнала либо вызов службы syslog. В таком случае нужно вести наблюдение за процессом, чтобы прекратить дальнейшие попытки запуска после нескольких неудач. В любой Linux-системе существует утилита, которая может делать это автоматически: init. Та же программа, что отвечает за загрузку системы и настройку терминалов, подходит для проверки постоянной работы программ — это и есть ее основное назначение.

Добавьте в файл *etc/inittab* произвольные строки, задающие программы, за которыми должен следить сценарий *init*:

```
zz:12345:respawn:/usr/local/sbin/my_daemon
```

Строка inittab состоит из идентификатора (уникальное сочетание двух произвольных символов — в нашем случае это zz), за которым следует список уровней запуска для данной программы, ключевое слово respawn и полный путь к нужной команде. В этом примере, поскольку

команда my_daemon сконфигурирована для запуска в фоновом режиме, сценарий init породит новую копию независимо от существования предыдущей. После внесения изменений в inittab не забудьте выполнить команду HUP (Перечитать файл) процессу init, после чего тот должен повторно загрузить измененные настройки. Один из быстрых способов сделать это таков:

```
# kill -HUP 1
```

Если новые команды будут порождаться слишком быстро, программа init отложит их выполнение на некоторое время, чтобы не допустить поглощения всех доступных ресурсов. Например:

```
zz:12345:respawn:/bin/touch /tmp/timestamp
```

В результате к файлу */tmp/timestamp* обращение будет произведено несколько раз за секунду, пока сценарий init не решит, что этого достаточно. Практически сразу в журнале */var/log/messages* можно будет увидеть следующее сообщение:

```
Sep 8 11:28:23 catlin init: Id "zz" respawning too fast: disabled for 5 minutes
```

Через пять минут программа init попытается запустить данную команду повторно, и если ситуация повторится, ее запуск будет снова отложен.

Этот метод хорошо подходит для выполнения команд с привилегиями пользователя root, но что если вам понадобится автоматически порождать процессы от имени другого пользователя? Никаких проблем: в этом поможет программа sudo:

```
zz:12345:respawn:/usr/bin/sudo -u rob /bin/touch /tmp/timestamp
```

Обратите внимание: команда touch будет запущена от имени пользователя rob, а не root. Если вы выполняете примеры в том же порядке, в каком они следуют в тексте книги, не забудьте перед запуском команды sudo удалить существующий файл */tmp/timestamp*. Затем, отправив команду HUP программе init, взгляните на файл *timestamp*:

```
rob@catlin:~# ls -al /tmp/timestamp
-rw-r--r--1 rob users 0 Sep 8 11:28 /tmp/timestamp
```

Применение программы init для запуска необходимых вам процессов имеет два недостатка: во-первых, придется закомментировать соответствующую строку в файле *inittab*, чтобы отключить процесс (поскольку если просто запустить команду kill для его уничтожения, процесс будет порожден заново), а во-вторых, вносить изменения в файл *inittab* может только пользователь root. Но если целью является сохранение процесса в рабочем состоянии при любых условиях, тогда программа init – это именно то, что требуется.

Дополнительная информация.

- *«Заставим программу sudo работать не покладая рук»* **[Совет № 12].**

СОВЕТ №5 n>&m: перенаправление стандартных выходных данных и стандартных ошибок

(Перенаправление стандартных выходных данных и стандартных ошибок на произвольное устройство вывода информации)

По умолчанию сообщения о стандартных ошибках команд направляются на ваш терминал. Стандартные выходные данные могут выводиться на терминал либо перенаправляться куда-либо еще (в файл, канал (pipe), либо заключаться в обратные кавычки (backticks)).

Иногда может возникнуть необходимость в выполнении обратной процедуры. Например, отправить стандартные выходные данные на экран и сохранить сообщения об ошибках в обратных кавычках, либо перенаправить стандартные выходные данные в файл, а стандартные сообщения об ошибках в канал, в команду обработки ошибок. Рассмотрим реализацию этой процедуры в рамках оболочки Bourne, т.к. оболочка C делать это не умеет.

Дескрипторы файлов 0, 1 и 2 соответствуют стандартным входным данным, стандартным выходным данным и стандартным ошибкам, соответственно. По умолчанию они связаны с файлом терминала */dev/tty*. Можно перенаправить любой дескриптор в произвольный файл — если, конечно, известно его имя. Чтобы перенаправить, к примеру, файловый дескриптор в файл *errfile*, наберите в командной строке:

```
$ имя_команды 2> errfile
```

Перенаправление стандартных выходных данных организовано также через канал и обратные кавычки:

```
$ имя_команды |...\
$ var='имя_команды'
```

Но, поскольку с каналом либо обратными кавычками не связано имя файла, нельзя воспользоваться перенаправлением >2 и придется переназначить файловые дескрипторы, не зная, какие файлы (и не только файлы) с ними связаны. Далее поясняется, как это сделать.

Начнем издалека, с отправки стандартных выходных данных и стандартных ошибок в канал или обратные кавычки. Изменять файлы и файловые дескрипторы можно при помощи оператора n>&m оболочки Bourne. Действия, выполняемые данным оператором, можно описать как «сделать так, чтобы файловый дескриптор n указывал на тот же файл, что и файловый дескриптор m». Воспользуемся этим

оператором для предыдущего примера, настроив пересылку стандартных ошибок туда же, куда мы направили стандартные выходные данные:

```
$ имя_команды 2>&1 |...
$ var='имя_команды 2>&1'
```

В обоих примерах оператор 2>&1 можно истолковать как «пересылать стандартные ошибки (файловый дескриптор 2) в то же самое место, что и стандартные выходные данные (файловый дескриптор 1)».

Можно использовать в одной строке не один оператор n>&m. Оболочка проводит грамматический разбор командной строки перед ее выполнением слева направо.

«Отлично! – скажете вы. – Значит, я могу поменять местами стандартные выходные данные и стандартные ошибки – направить stderr в канал, а stdout на экран!»

```
$ имя_команды 2>&1 1>&2 |... (неверно)
```

Увы! Проведя грамматический разбор строки 2>&1 1>&2, оболочка вначале выполнит оператор 2>&1. Мы уже видели результаты предыдущего примера — файловый дескриптор 2 (stderr) перенаправит стандартные ошибки в то же самое место, что и файловый дескриптор 1 (stdout). Только затем оболочка выполнит оператор 1>&2. В итоге дескриптор stdout (1) перенаправит стандартные выходные данные туда же, куда указывает дескриптор stderr (2), который, однако, и так уже ссылается в то же место, что и дескриптор stdout, а именно: в канал.

Это именно тот случай, когда возникает необходимость в использовании дополнительных дескрипторов, с 3 по 9. Ими пользуются довольно редко, и один из них можно использовать в качестве «ячейки памяти», хранящей информацию о том, куда ссылается другой дескриптор. Например, оператор 3>&2 можно истолковать как «заставить дескриптор 3 ссылаться на то же место, что и дескриптор 2». После задания этого оператора настройте дескриптор 2 так, чтобы он указывал на какое-то другое место. Затем настройте файловый дескриптор 1, чтобы он ссылался туда, куда раньше указывал файловый дескриптор 2 (и куда теперь указывает файловый дескриптор 3).

В таком случае понадобится набрать следующую команду:

```
$ имя_команды 3>&2 2>&1 1>&3 |...
$ var='имя_команды 3>&2 2>&1 1>&3'
```

Все открытые файлы закрываются автоматически по завершении процесса. Однако безопаснее закрывать эти файлы вручную. В этом случае, если вы забудете и воспользуетесь этим же дескриптором позд-

нее для чего-нибудь еще, например, обратитесь к файловому дескриптору 3 для перенаправления какой-то другой команды либо тот же дескриптор будет вызван вложенным процессом, то защитите себя от возникновения конфликтных ситуаций. Чтобы закрыть файловый дескриптор входных данных m необходимо выполнить оператор m<&-, а чтобы закрыть файловый дескриптор выходных данных m — оператор m>&-. То есть, для закрытия стандартных входных данных необходимо использовать оператор <&-; оператор >&-, в свою очередь, закрывает стандартные выходные данные.

СОВЕТ № 6

Составление сложных командных строк

(Вставка простых команд в законченные абзацы для создания составных отчетов.)

Изучение системы Linux, как и любой разновидности ОС Unix, во многом напоминает изучение иностранного языка. На определенном этапе обучения, после того как вы научились произносить простые односложные слова, необходимо начинать построение связных общеупотребительных фраз. В конце концов, вы обнаружите в себе способность выражаться полноценными предложениями на языке Unix и сможете сосредоточиться исключительно на решении возникающих проблем, а не на синтаксисе той или иной команды. Но, подобно тому как студенты, обучающиеся иностранному языку в высшей школе, вначале тратят большую часть времени на вопросы «как пройти туда-то?» и разъяснения по поводу того, что собой представляет дательный падеж, так и путь к беглому владению языком командной строки Linux начинается с изучения первых ключевых слов.

Оболочка командной строки достаточно терпелива и способна «выслушивать» каждое ваше высказывание, пока оно не обретет законченную форму. Любая команда может выступать в качестве входных данных для некоторой другой команды, позволяя создавать в Unix довольно интересные «предложения». Воспользовавшись клавишей «стрелка вверх», можно связать в одну цепочку ранее введенные команды для реализации сложного поведения.

Предположим, что перед вами стоит задача выяснить, почему через определенные промежутки времени веб-сервер вызывает целый ряд ошибок. Если вы наберете в командной строке less error_log, то увидите список ошибок, связанных с пропущенными графическими файлами (либо неправильно заданными ссылками на них):

```
[Tue Aug 27 00:22:38 2002] [error] [client 17.136.12.171] File does not
exist:/htdocs/images/spacer.gif
```

```
[Tue Aug 27 00:31:14 2002] [error] [client 95.168.19.34] File does not
exist: /htdocs/images/trans.gif
[Tue Aug 27 00:36:57 2002] [error] [client 2.188.2.75] File does not
exist: /htdocs/images/linux/arrows-linux-back.gif
[Tue Aug 27 00:40:37 2002] [error] [client 2.188.2.75] File does not
exist: /htdocs/images/linux/arrows-linux-back.gif
[Tue Aug 27 00:41:43 2002] [error] [client 6.93.4.85] File does not exist:
/htdocs/images/linux/hub-linux.jpg
[Tue Aug 27 00:41:44 2002] [error] [client 6.93.4.85] File does not exist:
/htdocs/images/linux/hub-xml.jpg
[Tue Aug 27 00:42:13 2002] [error] [client 6.93.4.85] File does not exist:
/htdocs/images/linux/hub-linux.jpg
[Tue Aug 27 00:42:13 2002] [error] [client 6.93.4.85] File does not exist:
/htdocs/images/linux/hub-xml.jpg
```

и т.д. Запустив пакет для ведения журнала, такой, как, например, analog, вы узнаете, сколько именно ошибок произошло за день, и даже некоторые подробности (что позволяет выявлять проблемы, требующие первоочередного вмешательства). Если захотите просмотреть файл журнала напрямую, то сможете почерпнуть из него мельчайшие детали, однако в целом этот файл содержит слишком много информации для эффективной обработки.

Начнем с простого. Существуют ли другие ошибки, не связанные с отсутствующими файлами? Во-первых, необходимо выяснить, сколько всего ошибок имело место за сегодняшний день:

```
$ wc -l error_log
1265 error_log
```

Теперь выясним, сколько ошибок связано с отсутствующими файлами:

```
$ grep "File does not exist:" error_log |wc -1
1265 error_log
```

Хорошее начало. Теперь, по крайней мере, известно, что других проблем или ошибок, связанных с генерацией динамического содержимого, типа ошибок сценариев cgi, нет. Если ошибки связаны с отсутствующими файлами либо опечатками в ссылках внутри HTML-кода, то ситуация не является критичной. Сформируйте список имен файлов, которые не были найдены. Нажмите клавишу «стрелка вверх» и отредактируйте предыдущую команду следующим образом:

```
$ grep "File does not exist:" error_log | awk '{print $13}' | less
```

Это именно то, что нужно (в 13-м поле содержится имя файла), но секундочку! Одни и те же имена файлов повторяются в этом списке много раз. Отфильтруйте список так, чтобы каждое имя файла встречалось только один раз:

```
$ grep "File does not exist:" error_log | awk '{print $13}' | sort | uniq | less
```

Замените оператор less на wc -l, чтобы узнать количество отсутствующих файлов. Но это не решает проблему в целом. Возможно, к одному из этих файлов было произведено только одно обращение, тогда как другой файл запрашивался несколько сотен раз. Как правило, если где-то в ссылке присутствует опечатка, впоследствии мы сталкиваемся со множеством запросов к одному и тому же файлу. Однако команда не дает никакого представления о том, к какому из файлов было произведено наибольшее количество обращений. Это решается с помощью оболочки командной строки bash; достаточно воспользоваться командной строкой для создания цикла for:

```
$ for x in 'grep "File does not exist:" error_log | awk '{print $13}' | sort | uniq'; do \
echo -n "$x : "; grep $x error_log | wc -l; done
```

Используйте обратные кавычки (’), чтобы полностью выполнить команду из предыдущего примера, перенаправив результаты ее выполнения в цикл for. При каждой итерации переменная $x задает переход к следующей строке выходных данных оригинальной команды, т.е. к следующему уникальному имени отсутствующего файла. Примените команду grep для данного имени файла в журнале error_log и подсчитайте, сколько раз эта строка встречается в файле журнала. Команда echo, в свою очередь, выведет результаты на дисплей в условно удобном для чтения формате.

Выводимые результаты условно удобны для чтения, т.к. на экран выводится информация по однократно встречающимся ошибкам (будем считать, что они нас в данном случае мало интересуют), результаты не отформатированы и даже не отсортированы! Задайте сортировку по количеству ошибок, связанных с определенным именем файла, благодаря чему файлы, к которым было произведено наибольшее количество обращений, окажутся в начале списка, а сам список ограничьте двадцатью наиболее часто встречающимися файлами:

```
$ for x in 'grep "File does not exist:" error_log | awk '{print $13}' | sort | uniq'; do\
grep $x error_log | wc -l | tr -d '\n';
echo " : Sx"; done | sort +2 -rn | head -20
```

Такой формат намного удобнее для чтения и анализа, а чтобы набрать эту команду в командной строке, понадобится времени ненамного больше, чем в предыдущем примере. Команда tr необходима для включения символа возврата каретки после символов новой строки в результатах, возвращенных командой wc (почему в самой команде отсутствует такой переключатель, непонятно). Результат будет выглядеть приблизительно так:

```
595: /htdocs/images/pixel-onlamp.gif.gif
156: /htdocs/image/trans.gif
```

```
139: /htdocs/images/linux/arrows-linux-back.gif
68: /htdocs/pub/a/onjava/javacook/images/spacer.gif
50: /htdocs/javascript/2001/03/23/examples/targer.gif
```

Из этого отчета видно, что почти половина ошибок связана с опечаткой в ссылке на веб-ресурс (обратите внимание на .gif.gif в конце первой строки). Следующая ошибка связана с опечаткой в пути (скорее всего, имелась в виду папка /images, а не /image). С остальными ошибками придется разбираться группе обслуживания веб-сервера, которой мы вышлем отчет по электронной почте:

```
$ (echo "Here's a report of the top 20 'missing' files in the error_log.";
echo; \
$ for x in 'grep "File does not exist:" error_log | awk '{print $13}' | sort | uniq'; do \
grep $x error_log | wc -l | tr -d '\n';
echo " : Sx"; done | sort +2 -rn | head -20 | mail -s "Missing file
report" webmaster@oreillynet.com
```

а одну твердую копию, возможно, придется распечатать для еженедельной встречи разработчиков:

```
$ for x in 'grep "File does not exist:" error_log | awk '{print $13}' | sort | uniq'; do \
grep $x error_log | wc -l | tr -d '\n';
echo " : Sx"; done | sort +2 -rn | head -20 | enscript
```

Хакерский совет для хакерского совета

Когда вы привыкнете использовать группы команд, то сможете объединять в цепочку результаты их выполнения, создавая любые виды отчетов, необходимых для формирования активного потока данных. При необходимости постоянного выполнения некоторого процесса можно написать для него отдельный сценарий оболочки (либо для большей эффективности реализовать сценарий на языке Perl или Python, поскольку использование в командной строке | означает порождение другой программы). На современных машинах разница вряд ли будет заметна, однако написание сценария считается более правильной формой решения проблемы. А в командной строке достаточно места для реализации самых разнообразных хакерских приемов.

СОВЕТ №7

Работа с файлами при помощи утилиты xargs

(Обработка групп файлов, в именах которых содержатся пробелы и другие запрещенные символы)

Если имена файлов содержат пробелы, круглые скобки и другие запрещенные символы, могут возникнуть сложности с их обработкой. Пользователи сталкиваются с этой проблемой все чаще, особенно

в связи со взрывным ростом популярности цифровой музыки. К счастью, в оболочке bash для обработки отдельных файлов предусмотрено использование завершающей табуляции. Например:

```
rob@catlin:~/Music$ ls
Hallucinogen – The Lone Deranger
Misc – Pure Disco
rob@catlin:~/Music$ rm -rf Misc[Tab]rob@catlin:~/Music$ rm -rf Misc\ -\ Pure\
Disco/
```

Нажатие клавиши табуляции TAB в конце строки производит замену одной командной строки на другую, приведенную после символа табуляции, в результате чего корректно обрабатываются все специальные символы, содержащиеся в имени файла. Этот прием хорош для обработки одного файла. А если необходимо выполнить обработку нескольких файлов, например, переименовать группу музыкальных файлов так, чтобы включить в имя файла название альбома? Взгляните на следующий пример:

```
rob@catlin:~/Music$ cd Hall[TAB]
rob@catlin:~/Music$ cd Hallucinogen\ -\ The\ Lone\ Deranger/
rob@catlin:~/Music/Hallucinogen – The Lone Deranger$ ls
Hallucinogen – 01 – Demention.mp3
Hallucinogen – 02 – Snakey Shaker.mp3
Hallucinogen – 03 – Trancespotter.mp3
Hallucinogen – 04 – Horrorgram.mp3
Hallucinogen – 05 – Snarling (Remix).mp3
Hallucinogen – 06 – Gamma Goblins Pt. 2.mp3
Hallucinogen – 07 – Deranger.mp3
Hallucinogen – 08 – Jiggle of the Sphinx.mp3
rob@catlin:~/Music/Hallucinogen – The Lone Deranger$
```

При попытке одновременной манипуляции несколькими файлами ситуация немного усложнится. Большинство системных утилит неправильно обрабатывают пробельные символы, выдавая избыточное количество порций данных, либо завершают свою работу с ошибкой, если встретят в имени файла какие-либо скобки: () или {}. Следовательно, нужен такой разделитель, который гарантированно не будет встречаться в имени файла.

Утилита xargs позволяет преодолевать проблемы с NULL-символами. Рассмотрим следующий сценарий.

Листинг: albumize

```
#!/bin/sh
if [ -z "$ALBUM" ]; then
echo 'You must set the ALBUM name first (eg. export ALBUM="Greatest Hits")'
```

```
# Вначале необходимо задать название альбома
#(например: ALBUM="лучшие хиты")
exit 1
fi
for x in *; do
echo -n $x; echo -ne '\000'
echo -n `echo $x|cut -f 1 -d '-'`
echo -n " - $ALBUM - "
echo -n `echo $x|cut -f 2- -d '-'`; echo -ne '\000'
done | xargs -0 -n2 mv
```

Здесь кроется двойная хитрость. Во-первых, создан список оригинальных имен файлов, за которыми будут следовать новые имена, присваиваемые этим файлам при помощи команды mv и разделенные NULL-символами для всех файлов в текущей директории. Затем этот список передается программе xargs с двумя переключателями: переключатель -0 означает, что в качестве разделителей используются NULL-символы (вместо пробельных символов и символов новой строки), а переключатель -n2 означает, что за один раз команде mv будут передаваться два аргумента.

Сохраните сценарий под названием *~/bin/albumize*. Перед его запуском назначьте переменной среды $ALBUM строку символов, которую хотите добавить в имя файла сразу после первого тире (-). Рассмотрим пример:

```
rob@catlin:~/Music/Hallucinogen - The Lone Deranger$ export ALBUM="The Lone
Deranger"
rob@catlin:~/Music/Hallucinogen - The Lone Deranger$ albumize
rob@catlin:~/Music/Hallucinogen - The Lone Deranger$ ls
Hallucinogen - The Lone Deranger - 01 - Demention.mp3
Hallucinogen - The Lone Deranger - 02 - Snakey Shaker.mp3
Hallucinogen - The Lone Deranger - 03 - Trancespotter.mp3
Hallucinogen - The Lone Deranger - 04 - Horrorgram.mp3
Hallucinogen - The Lone Deranger - 05 - Snarling (Remix).mp3
Hallucinogen - The Lone Deranger - 06 - Gamma Goblins Pt. 2.mp3
Hallucinogen - The Lone Deranger - 07 - Deranger.mp3
Hallucinogen - The Lone Deranger - 08 - Jiggle of the Sphinx.mp3
rob@catlin:~/Music/Hallucinogen - The Lone Deranger$
```

Что нужно сделать, чтобы удалить название альбома из имен файлов? Попробуйте написать следующий сценарий, сохранив его под именем *~/bin/dealbumize*:

```
#!/bin/sh
for x in *; do
echo -n $x; echo -ne '\000'
echo -n `echo $x|cut -f 1 -d '-'`; echo -n ' - '
echo -n `echo $x|cut -f 3- -d '-'`; echo -ne '\000'
done | xargs -0 -n2 mv
```

и запустите его:

```
rob@catlin:~/Music/Hallucinogen – The Lone Deranger$ dealbumize
rob@catlin:~/Music/Hallucinogen – The Lone Deranger$ ls
Hallucinogen – 01 – Demention.mp3
Hallucinogen – 02 – Snakey Shaker.mp3
Hallucinogen – 03 – Trancespotter.mp3
Hallucinogen – 04 – Horrorgram.mp3
Hallucinogen – 05 – Snarling (Remix).mp3
Hallucinogen – 06 – Gamma Goblins Pt. 2.mp3
Hallucinogen – 07 – Deranger.mp3
Hallucinogen – 08 – Jiggle of the Sphinx.mp3
rob@catlin:~/Music/Hallucinogen – The Lone Deranger$
```

Переключатель -0 часто используется в сочетании с опцией поиска -print0 (которая обычно выводит имена файлов, разделенные NULL-символами вместо символов новой строки). Применив к каналу команду find или xargs, вы можете делать все, что захотите, с любым количеством файлов, и при этом нет угрозы ошибки, связанной со слишком длинным списком аргументов:

```
rob@catlin:~/Pit of too many files$ ls
bash: /bin/ls: Argument list too long
```

Сочетание команд find/xargs позволяет выполнять быструю обработку вышеупомянутых файлов, независимо от их названия:

```
rob@catlin:~/Pit of too many files$ find -type f -print0 | xargs -0 ls
```

Для удаления этих файлов просто замените завершающую команду ls на rm.

СОВЕТ № 8

Неизменяемые файлы в ext2/ext3

(Создание файлов, которыми не сможет оперировать даже пользователь root)

Предположим, вам понадобилось очистить директорию */tmp* и вдруг вы столкнулись со следующими сообщениями об ошибках:

```
root@catlin:/tmp# rm -rf junk/
rm: cannot unlink `junk/stubborn.txt`: Operation not permitted
rm: cannot remove directory `junk': Directory not empty
root@catlin:/tmp# cd junk/
root@catlin:/tmp/junk# ls -al
total 40
drwxr-xr-x 2 root root 4096 Sep 4 14:45 ./
drwxrwxrwt 13 root root 4096 Sep 4 14:45 ../
-rw-r--r--1 root root 4096 Sep 4 14:43 stubborn.txt
root@catlin:/tmp/junk# rm ./stubborn.txt
rm: remove write-protected file `./stubborn.txt'? y
rm: cannot unlink `./stubborn.txt': Operation not permitted
```

Что же это такое? Разве вы не являетесь пользователем root? Попытаемся очистить содержимое файла, вместо того чтобы удалить его:

```
root@catlin:/tmp/junk# cp /dev/null stubborn.txt
cp: cannot create regular file `stubborn.txt': Permission denied
root@catlin:/tmp/junk# > stubborn.txt
bash: stubborn.txt: Permission denied
```

Ведь очевидно, что папка */tmp* не может быть установлена только для чтения. В чем же тогда проблема?

В файловых системах ext2 и ext3, помимо стандартных битов, для файлов существует ряд дополнительных атрибутов, доступ к которым можно получить при помощи утилиты chmod. Если вам до сих пор не приходилось с ними сталкиваться, обратитесь к страницам руководства по команде chattr и ее аналогу, команде lsattr.

Одним из весьма полезных новых атрибутов является флажок -i (immutable – неизменяемый). Если этот атрибут задан для определенного файла, все попытки его переименования, удаления ссылок на него, перезаписи либо присоединения к файлу дополнительной информации запрещены. Запрещено даже создание жестких ссылок (hard links), т.е. нельзя их создать, а затем отредактировать. И даже наличие привилегий пользователя root не позволяет получать доступ к файлу:

```
rob@catlin:~/tmp/junk# ln stubborn.txt another.txt
ln: creating hard link 'another.txt' to 'stubborn.txt': Operation not
permitted
```

Чтобы просмотреть заданные для файла вспомогательные флажки файловой системы ext, воспользуйтесь командой lsattr:

```
rob@catlin:~/tmp/junk# lsattr
----i---------- .stubborn.txt
```

а для установки флажков аналогично команде chmod воспользуйтесь командой chattr:

```
rob@catlin:~/tmp/junk# chattr -i stubborn.txt
rob@catlin:~/tmp/junk# rm stubborn.txt
rob@catlin:~/tmp/junk#
```

Эта функция обеспечивает дополнительную защиту файлов, в которые вы не собираетесь когда-либо вносить изменения, например файлы */etc/rc.d/** либо различные файлы конфигурации. Неизменяемые файлы перестанут быть доступной жертвой атак путем перезаписи со стороны других процессов, даже если те запущены от имени пользователя root.

Существуют также флажки сжатия, безопасного удаления, невозможности удаления, синхронной записи и др. На момент написания данной

книги многие дополнительные атрибуты еще не были реализованы, поэтому следите за новыми разработками в области файловой системы ext.

СОВЕТ № 9

Ускорение компиляции

(Убедитесь, что ни один из процессоров не простаивает)

Если вы работаете на многопроцессорной системе (SMP), обладающей средними объемами ОЗУ, можно значительно выиграть в производительности, если задать в команде make параллельное выполнение компиляции, получив тем самым значительный прирост производительности по сравнению с последовательной компиляцией, используемой по умолчанию. Чтобы разрешить во время компиляции выполнение более чем одного дочернего процесса, воспользуйтесь переключателем -j:

```
rob@mouse:~/linux$ make -j4; make -j4 modules
```

Некоторые проекты просто не предназначены для параллельной компиляции, поэтому, если какие-то части проекта будут созданы раньше, чем закончится компиляция родительских объектов, от которых они зависят, возможно возникновение проблем. Начните компиляцию сначала, удалив из командной строки флажок -j.

Ниже приведен ряд тестов на скорость компиляции. Они проводились на двухпроцессорной системе с двумя PIII/600 МГц с 1 ГБ ОЗУ. В каждом случае выполнялась компиляция программы bzImage для Linux 2.4.19 (стандартные выходные данные STDOUT перенаправлялись на устройство */dev/null*), а перед началом каждой новой попыткой дерево исходных кодов удалялось.

time make bzImage:

```
real 7m1.640s
user 6m44.710s
sys 0m25.260s
```

time make -j2 bzImage:

```
real 3m43.126s
user 6m48.080s
sys 0m26.420s
```

time make -j4 bzImage:

```
real 3m37.687s
user 6m44.980s
sys 0m26.350s
```

time make -j10 bzImage:

```
real 3m46.060s
user 6m53.970s
sys 0m27.240s
```

Как видите, налицо существенное ускорение компиляции при добавлении переключателя -j2. Реальное время выполнения команды make сократилось с 7 минут до 3 минут 43 секунд. Задание переключателя -j4 позволило сократить время компиляции еще на несколько секунд, но при использовании флажка -j10 выполнение компиляции, наоборот, на несколько секунд замедлилось. Обратите внимание: временные показатели для user и sys практически не отличаются во всех четырех случаях. В конце концов, вам нужно обработать одно и то же количество бит, просто с помощью флажка -j вы можете поделить эту задачу между несколькими процессорами.

СОВЕТ №10 Как чувствовать себя комфортно в оболочке командной строки

(Переменные среды делают работу с оболочкой bash более удобной)

Чтение оперативного руководства по оболочке bash может занять много времени, особенно если вы не знаете точно, что вам нужно искать. Но когда вы найдете время, которое сможете посвятить его чтению, — обязательно прочтите. Эта оболочка богата на различные скрытые, но действительно полезные функции, большинство которых просто отключено по умолчанию.

Начнем с рассмотрения некоторых полезных переменных среды и значений, которые могут быть для них заданы:

```
export PS1='echo -ne "\033[0;34m\u@\h:\033[0;36m\w\033[0;34m\$\033[0;
37m "'
```

Как известно, переменная PS1 отвечает за настройку приглашения командной строки системы по умолчанию и автоматическую интерпретацию ESC-последовательностей, таких как \u (для имени пользователя) или \w (для текущей рабочей директории). О чем вы можете не знать, так это о возможности закодировать ESC-последовательности ANSI в приглашении командной строки для «раскраски» выводимых данных. Мы заключили всю строку в одинарные кавычки (’), чтобы заставить команду echo генерировать магический ESC-символ ASCII. Данная команда выполняется один раз, после чего результат ее выполнения сохраняется в переменной PS1. Еще раз внимательно посмотрите на данную строку, выделив жирным шрифтом все, что не относится к коду ANSI:

```
export PS1='echo -ne "\033[0;34m\u@\h:\033[0;36m\w\033[0;34m\$\033[0;
37m "'
```

Нетрудно узнать приглашение командной строки \u@\h:\w\$, которое все хорошо знают. Меняя числа, следующие после каждого двоеточия, можно задать цвета для каждой части приглашения командной строки, а также команду, которая будет выполняться каждый раз перед выводом приглашения командной строки оболочкой bash:

```
export PROMT_COMMAND='echo -ne "\033]0;${USER}@${HOSTNAME}: ${PWD}\007"'
```

Здесь одинарные кавычки не нужны, поскольку в кавычках оболочка bash ожидает увидеть реальную команду, а не строку символов. ESC-последовательность позволяет менять строку заголовка в большинстве окон терминала (таких как xterm, rxvt, eterm, gnometerm и т.д.). Все символы, размещенные между точкой с запятой и \007, будут добавляться в строку заголовка каждый раз при обновлении приглашения командной строки. В данном случае на экран выводятся имя пользователя, имя компьютера и текущая рабочая папка. Это весьма удобно для того, чтобы определить (даже из утилиты vim), к какому компьютеру вы подключены и в какую директорию собираетесь сохранить файл. В разделе *«Постоянное отображение средней загрузки системы в строке заголовка»* **[Совет № 59]** показано, как обновлять строку заголовка в режиме реального времени, а не при повторном выводе приглашения командной строки оболочки bash.

Вы когда-нибудь случайно нажимали сочетание клавиш ^D в одной строке слишком много раз, в результате чего оказывались выгруженными из оболочки командной строки? Можно заставить оболочку bash игнорировать столько последовательных нажатий ^D, сколько пожелаете:

```
export IGNOREEOF=2
```

В итоге оболочка начнет придерживаться правила Снарка «То, что я сказал тебе трижды — правда» и выгрузит вас как пользователя только в том случае, если в одной строке будет трижды нажато ^D. Если для вас это значение слишком мало, можете задать 101 раз, а оболочка bash будет тщательно подсчитывать количество нажатий.

Очень удобно иметь внутри своего домашнего каталога директорию, упомянутую в переменной path (для хранения сценариев, символических ссылок и других случайных фрагментов кода). Традиционно в качестве данной директории выступает папка *bin*. Если воспользоваться в оболочке командной строки bash опцией ~, например,

```
export PATH=$PATH:~/bin
```

путь к ней будет всегда указываться правильно, даже если вы захотите переместить домашний каталог либо использовать эту строку сцена-

рия на разных компьютерах, на которых домашние каталоги потенциально могут быть размещены в разных местах — как в сценарии movein.sh (см. раздел *«Быстрый перенос настроек при помощи movein.sh»* **[Совет № 72]**).

Подобно тому как осуществляется поиск команд по каталогам, перечисленным в переменной PATH, и справочным руководствам по каталогам, перечисленным в переменной MANPATH, при запуске команды cd выполняется поиск каталогов по переменной CDPATH. По умолчанию она принимает значение ".", но по желанию его можно изменить:

```
export CDPATH.:~
```

В результате поиск с помощью команды cd будет выполняться не только в текущем каталоге, но и в вашем домашнем каталоге и директории, к которой вы собираетесь перейти. Например:

```
rob@caligula:~$ ls
bin/ devel/ incoming/ mail/ test/ stuff.txt
rob@caligula:~$ cd /usr/local/bin
rob@caligula:/usr/local/bin$ cd mail
bash: cd: mail: No such file or directory
rob@caligula:/usr/local/bin$ export CDPATH=.:~
rob@caligula:/usr/local/bin$ cd mail
/home/rob/mail
rob@caligula:~/mail$
```

В переменной CDPATH можно задать столько каталогов, сколько пожелаете, использовав в качестве разделителя символ : (аналогично тому, как это делается в переменных PATH и MANPATH).

Мы знаем практически все о функции, вызываемой нажатием клавиши «стрелка вверх», и команде history. Но что произойдет, если случайно напечатать в командной строке некоторую критическую информацию? Предположим, вы ошиблись и вместо команды набрали пароль. Этот пароль сохранится в файле журнала команд *~/.bash_history* и останется там даже после выхода из системы, откуда он может быть извлечен каким-нибудь беспринципным пользователем. Редактирование файла */.bash_history* вручную также не принесет желаемого результата, поскольку этот файл перезаписывается каждый раз при выходе из системы. Для быстрой очистки истории команд попробуйте запустить из командной строки следующую команду:

```
export HISTSIZE=0
```

В итоге файл истории команд будет обнулен и при выходе из системы останется пустым. При повторном входе в систему ведение истории

команд будет продолжено в обычном режиме. С этого момента постарайтесь вести себя более осмотрительно.

Вы когда-нибудь сталкивались с проблемой, когда пользователи подключаются к системе, а затем отключают свой ноутбук и уходят домой, забыв выйти из системы? Если вы когда-нибудь запускали службу w и наблюдали кучу пользователей, подключенных к системе уже в течение нескольких дней, попробуйте задать в своем окружении следующую переменную:

```
export TMOUT=600
```

В переменной TMOUT задается количество секунд, по истечении которых оболочка bash автоматически выгрузит пользователя из системы, если в командной строке не было введено ни одной команды. Такой прием не сработает, если ваши пользователи находятся в окне редактора vi, но в случае, когда они работают только с командной строкой, проблему действительно можно решить подобным образом. Для некоторых пользователей, разумеется, 10 минут могут оказаться слишком маленьким промежутком времени, но им можно вежливо напомнить, что если их не устраивают значения системных переменных по умолчанию, они могут изменить их по своему усмотрению.

Здесь возникает один интересный вопрос: как сделать внесенные изменения постоянными? Оболочка командной строки bash во время загрузки считывает настройки из разных файлов в зависимости от того, как она была вызвана: при входе пользователя в систему либо путем запуска из другой оболочки. В первом случае, т.е. при запуске оболочки во время входа в систему bash(1):

> ... в первую очередь происходит обращение и выполнение команд из файла */etc/profile*, при наличии такового. Затем выполняется поиск файлов *~/.bash_profile*, *~/.bash_login* и *~/.profile*, именно в таком порядке осуществляется считывание и выполнение команд, начиная с первого существующего и доступного для чтения файла... Если оболочка командной строки существует, то bash считывает и выполняет команды, хранимые в файле *~/.bash_logout*, если таковой существует.

При вызове из других оболочек:

> оболочка командной строки bash считывает и выполняет команды из файла *~/.bashrc*, если таковой существует.

Полное определение того, что понимается под оболочкой входа в систему, а также исчерпывающая информация по окружению, в котором вы работаете каждый день, содержится в справочных сведениях по bash(1).

СОВЕТ №11 Поиск и удаление файлов с установленными битами setuid/setgid

(Защита от потенциальной угрозы эксплуатации привилегий пользователя root)

Если в роли сервера выступает система под управлением ОС Linux, то главным является вопрос: «Чего, собственно, мы добиваемся?». Имеет ли смысл установка на веб-сервере подсистемы печати? Нужно ли на системе, в которой отсутствует консоль, устанавливать пакет gpm? Как правило, установка лишних программных пакетов приводит к напрасной трате дискового пространства, однако в случае наличия файлов с установленными битами идентификаторов setuid и setgid ситуация еще более усложняется.

Хотя разработчики трудятся не покладая рук, чтобы защитить уязвимые места, связанные с использованием этих битов, создается впечатление, что почти каждый месяц неожиданно обнаруживаются все новые уязвимые места. Вы должны регулярно проверять состояние файлов с установленными битами setuid и setgid на своей системе, особенно если оболочку командной строки сервера используете не только вы.

Ниже рассмотрена команда, выполняющая поиск всех файлов с установленным битом setuid или setgid:

```
root@catlin:~# find / -perm +6000 -type f -exec ls -ld {} \;
> setuid.txt &
```

В результате выполнения данной команды создается файл под названием *setuid.txt*, в который заносится информация обо всех обнаруженных в системе файлах, соответствующих заданному критерию. Внимательно изучите полученный список и удалите биты setuid и setgid со всех неиспользуемых вами утилит.

Посмотрим, какие файлы с установленными битами setuid или setgid можно найти на обычной системе:

```
-rws--x--x 1 root bin 35248 May 30 2001 /usr/bin/at
-rws--x--x 1 root bin 10592 May 30 2001 /usr/bin/crontab
```

Ничего удивительного. Утилиты at и crontab нуждаются в привилегиях пользователя root, чтобы была возможность запуска заданий в at или crontab от имени пользователей. Если вы не пользуетесь вышеупомянутыми службами, снимите с них биты setuid:

```
# chmod a-s /usr/bin{at,crontab}
```

Отключать планировщик — не самая лучшая идея, поскольку работа многих систем базируется на выполнении определенных заданий по расписанию. Но вспомните, когда вы в последний раз пользовались утилитой at? А ваши пользователи знают, для чего она предназначена?

Если на определенной системе она не нужна, имеет смысл просто от нее избавиться. Без бита setuid команда at будет недоступна для обычных пользователей, но в то же время с ней спокойно сможет работать пользователь root.

```
-rws--x--x 1 root bin 11244 Apr 15 2001 /usr/bin/disable-paste
```

Данная команда является частью пакета gpm (драйвера мыши для консоли Linux). Подключена ли к вашему компьютеру мышь? Используете ли вы ее в текстовом режиме? Если нет, тогда зачем оставлять включенным бит setuid для исполняемого файла, принадлежащего пользователю root, который никогда не будет запускаться?

```
-r-s--s--x 1 root lp 14632 Jun 18 2001 /usr/bin/lpq
-r-s--s--x 1 root lp 15788 Jun 18 2001 /usr/bin/lpr
-r-s--s--x 1 root lp 15456 Jun 18 2001 /usr/bin/lprm
-r-s--s--x 1 root lp 23772 Jun 18 2001 /usr/sbin/lpc
```

Все вышеперечисленные файлы относятся к подсистеме печати. Но действительно ли данная машина использует службу lp для вывода на печать?

```
-rws--x--x 1 root bin 33760 Jun 18 2000 /usr/bin/chage
-rws--x--x 1 root bin 29572 Jun 18 2000 /usr/bin/chfn
-rws--x--x 1 root bin 27188 Jun 18 2000 /usr/bin/chsh
-rws--x--x 1 root bin 35620 Jun 18 2000 /usr/bin/passwd
```

Эти файлы необходимы пользователям для задания паролей, оболочек и информации для службы finger. Используется ли на вашем вебсервере служба finger? Если, как на многих других сайтах, реестр пользователя хранится где-то в другом месте (скажем, в веб) и он не синхронизируется с полем GECOS, тогда вышеупомянутая информация почти бесполезна, за исключением «настоящего» имени пользователя, которое до сих пор используется некоторыми почтовыми клиентами. Вы действительно хотите позволить пользователям самостоятельно менять эти сведения, без вмешательства администратора?

```
-r-xr-sr-x 1 root tty 9768 Jun 21 2001 /usr/bin/wall
-r-xr-sr-x 1 root tty 8504 Jun 21 2001 /usr/bin/write
```

Для нормальной работы обеих утилит — wall и write — необходимо задать для них бит setgid, чтобы сделать возможным вывод на терминалы других пользователей. Как правило, это достаточно безопасная операция, однако даже ею могут воспользоваться злоумышленники с целью вывода некорректных данных (или *большого количества* некорректных данных) на терминалы других пользователей. Чтобы запретить другим пользователям выполнять подобные действия, отключите бит setgid.

Даже в этом случае пользователь root по-прежнему сможет отправлять команды wall и write (например, при отсылке сообщения в результате завершения работы во время перезагрузки системы).

```
-rwsr-xr-x 1 root bin 14204 Jun 3 2001 /usr/bin/rcp
-rwsr-xr-x 1 root bin 10524 Jun 3 2001 /usr/bin/rlogin
-r-sr-xr-x 1 root bin 7956 Jun 3 2001 /usr/bin/rsh
```

r-службы существовали еще до появления ssh, что было не так уж давно. Нужно ли предоставлять пользователям доступ к r-службам? Существует ли хоть какая-то функция, которую нельзя реализовать при помощи ssh и scp и для которой крайне необходимо воспользоваться службами rsh либо rcp? Скорее всего, немного поработав с утилитой ssh, вы перестанете испытывать ностальгию по r-службам и сможете избавиться от потенциальной бомбы с часовым механизмом в виде установленного бита setuid для исполняемого файла службы rsh. Если хотите знать, что интересного можно сделать при помощи ssh, найдите описание этой службы среди других советов, изложенных в этой книге.

```
-r-sr-xr-x 1 root bin 10200 Jun 3 2001 /usr/bin/traceroute
-r-sr-xr-x 1 root bin 15004 Jun 3 2001 /bin/ping
```

Бит setuid с целью назначения прав пользователя root обязательно должен быть задан для команд ping и traceroute, чтобы они могли создавать пакеты ICMP. Если хотите, чтобы ваши пользователи смогли запускать вышеупомянутые утилиты сетевой диагностики, тогда без данного бита просто невозможно обойтись. В противном случае снятие с этих файлов бита setuid ограничивает круг лиц, имеющих право на запуск команд ping и traceroute, единственным пользователем root.

```
-r-sr-sr-- 1 uucp uucp 83344 Feb 10 2001 /usr/bin/uucp
-r-sr-sr-- 1 uucp uucp 36172 Feb 10 2001 /usr/bin/uuname
-r-sr-sr-- 1 uucp uucp 93532 Feb 10 2001 /usr/bin/uustat
-r-sr-sr-- 1 uucp uucp 85348 Feb 10 2001 /usr/bin/uux
-r-sr-sr-- 1 uucp uucp 65492 Feb 10 2001 /usr/lib/uucp/uuchk
-r-sr-sr-- 1 uucp uucp 213832 Feb 10 2001 /usr/lib/uucp/uucico
-r-sr-sr-- 1 uucp uucp 70748 Feb 10 2001 /usr/lib/uucp/uuconv
-r-sr-sr-- 1 uucp uucp 315 Feb 10 2001 /usr/lib/uucp/uusched
-r-sr-sr-- 1 uucp uucp 95420 Feb 10 2001 /usr/lib/uucp/uuxqt
```

В наши дни постоянно работающих сетей TCP/IP UUCP применяется редко. Если UUCP не используетсся, оставление битов setuid и setgid включенными для его поддержки лишено всякого смысла.

Отключение ненужных привилегий позволяет минимизировать риск того, что когда-нибудь обнаруженное уязвимое место позволит осуществить эскалацию привилегий.

В этом разделе сделана попытка изложить материал путем рассмотрения процесса, а не готового решения. Но приведенный здесь список файлов никак нельзя считать законченным. Устанавливая сервер, необходимо помнить, в каких целях он будет использоваться, и действовать адекватным образом. Там, где это возможно, избавьтесь от лишних привилегий (либо даже целых пакетов), функциональность которых вам просто не нужна. Не стесняйтесь обращаться к оперативным руководствам, чтобы выяснить, за что отвечает данный исполняемый файл и зачем ему нужен бит setuid, а если оперативное руководство не сможет помочь, займитесь изучением исходных кодов.

СОВЕТ
№12

Заставим программу sudo работать не покладая рук

(Программа sudo перекладывает выполнение ваших поручений на плечи других пользователей без предоставления неограниченных полномочий на доступ к системе)

Утилита sudo позволяет пользователям выполнять некоторые функции, касающиеся администрирования системы, без предоставления полноценного доступа в качестве пользователя root. Этот исполняемый файл с установленным битом setuid пользователя root может запускать команды от имени авторизованного пользователя после указания текущего пароля.

Выступая в роли пользователя root, можно отредактировать список пользователей, которым позволено запускать программу sudo, в файле */usr/sbin/visudo*. По умолчанию список пользователей, которые могут выполнять эту программу, выглядит приблизительно так:

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

К сожалению, многие системные администраторы имеют тенденцию использовать эту строку в качестве шаблона, в одностороннем порядке предоставляя неограниченный доступ пользователя root для всех остальных администраторов.

```
root ALL=(ALL) ALL
rob ALL=(ALL) ALL
jim ALL=(ALL) ALL
david ALL=(ALL) ALL
```

Таким образом можно предоставить определенному человеку полномочия пользователя root, не сообщая ему пароль. Однако данный метод эффективен лишь в том случае, когда все пользователи программы sudo заслуживают полного доверия. При условии правильных настроек утилита sudo в состоянии дать огромную гибкость в плане предоставления доступа для выполнения любых команд под произвольным идентификатором пользователя (uid).

Синтаксис команды sudo таков:

```
пользователь хост =(фактический пользователь) команда
```

В первом столбце задается пользователь sudo. Во втором — определяются узлы сети, на которые распространяется действие данной команды. С ее помощью можно задавать идентичную конфигурацию для программы sudo на нескольких системах.

Предположим, к примеру, что у вас есть разработчик, который нуждается в доступе с правами пользователя root к своей системе, но не к какому-либо другому серверу:

```
peter beta.oreillynet.com=(ALL) ALL
```

В столбце, заключенном в круглые скобки, задается имя фактического пользователя, обладающего правами на запуск команды. Эта функция весьма удобна, чтобы разрешать одним пользователям выполнять программный код от имени других пользователей (необязательно root):

```
peter lists.oreillynet.com=(mailman) ALL
```

И, наконец, в последнем столбце перечислены команды, которые имеет право запускать пользователь:

```
david ns.oreillynet.com=(bind) /usr/sbin/rndc,/usr/sbin/named
```

Если необходимо задать длинный список команд либо, как в данном примере, список пользователей системы, воспользуйтесь в sudo командой alias. Псевдоним (alias) может применяться вместо соответствующей записи любой строки конфигурационного файла программы sudo:

```
User_Alias ADMINS=rob,jim,david
User_Alias WEBMASTERS=peter,nancy

Runas_Alias DAEMONS=bind,www,snmp,ircd

Host_Alias WEBSERVERS=www.oreillynet.com,www.oreilly.com,www.perl.com

Cmnd_Alias PROCS=/bin/kill,/bin/killall,/usr/bin/skill,/usr/bin/top
Cmnd_Alias APACHE=/usr/local/apache/bin/apachectl

WEBMASTERS WEBSERVERS=(www) APACHE
ADMINS ALL=(DAEMONS) ALL
```

Вместо отдельных пользователей можно также задавать целые системные группы, чтобы разрешить выполнение определенных команд всем пользователям, принадлежащим к той или иной группе. Для этого нужно добавить перед именем группы символ %:

```
%wwwadmin WEBSERVERS=(www) APACHE
```

Теперь любой пользователь, принадлежащий к группе wwwadmin, сможет запускать команду apachectl от имени пользователя www на любом веб-сервере.

Еще одной полезной функцией является флажок NOPASSWD:. Наличие данной опции означает, что пользователю не придется вводить пароль перед выполнением команды:

```
rob ALL=(ALL) NOPASSWD: PROCS
```

В результате пользователь rob сможет запускать команды kill, killall, skill и top на любом компьютере под учетной записью любого пользователя без ввода пароля. И, наконец, утилиту sudo можно считать удобной альтернативой программе su для автоматического запуска программ из системных файлов rc:

```
(cd /usr/local/mysql; sudo -u mysql ./bin/safe_mysql &)
sudo -u www /usr/local/apache/bin/apachectl start
```

Чтобы эта команда запускалась во время загрузки, необходимо наличие в файле конфигурации используемой по умолчанию строки root ALL=(ALL) ALL.

Используйте утилиту sudo с обычными предосторожностями, характерными для исполняемых файлов с установленным битом setuid. Необходимо четко представлять, что утилита позволяет выполнять произвольные команды от имени фактического пользователя, особенно если она применяется для выполнения интерактивных команд, например запуска редакторов, либо каких-то компиляторов или интерпретаторов. В большинстве случаев это не проблема, поскольку рассмотренный метод делегирования прав пользователям все равно предпочтительнее, чем предоставление неограниченного доступа с привилегиями пользователя root.

СОВЕТ №13 Применение утилиты Makefile для автоматизации задач администратора

(Утилита Makefile собирает быстро и просто все, а не только gcc)

Команда make чаще всего используется для компиляции проектов, нередко включающих базовые наборы компиляторов GNU (gcc), из исходных кодов. Но немногие представляют, что, поскольку утилита make отслеживает время модификации файлов, она неплохо подходит для выполнения разного рода обновлений, вызванных внесением изменений в определенные файлы.

Ниже рассматривается листинг сценария, используемого для обслуживания конфигурационных файлов программы sendmail.

Листинг: Makefile.mail

```
M4=     m4
CFDIR=  /usr/src/sendmail-8.12.5/cf
CHMOD=  chmod
ROMODE=444
RM=     rm -f

.SUFFIXES: .mc .cf

all: virtusers.db aliases.db access.db

access.db: access.txt
makemap -v hash access < access.txt

aliases.db: aliases
newaliases

virtusers.db: virtusers.txt
makemap -v hash virtusers < virtusers.txt

.mc.cf:
$(RM) $@
$(M4) ${CFDIR}/m4/cf.m4 $*.mc > $@ || ( $(RM) $@ && exit 1 )
$(CHMOD) $(ROMODE) $@
```

Поместив этот сценарий в файл */etc/mail/Makefile*, не нужно помнить о необходимости запуска команды newaliases после редактирования псевдонимов в файле *sendmail* или о синтаксисе команды makemap в ходе обновления виртуальных доменов либо доступа к настройкам управления. И самое замечательное заключается в том, что во время обновления ведущего файла настроек *mc* (вы ведь пользуетесь программой *mc*, а не редактируете файл *sandmail.cf* вручную, не так ли?) новый файл *.cf* будет создан автоматически. Поскольку она следит за последними обновлениями файлов, то выполняет перестройку только тех файлов, которые в этом нуждаются.

Приведем еще один пример, связанный с переносом файлов конфигурации Apache на другой сервер, например, при *циклической настройке сервера Apache*, о которой говорится в разделе *«Распределение нагрузки при помощи директивы RewriteMap сервера Apache»* **[Совет № 99]**. А пока просто поместите его в свою локальную директорию */usr/local/apache/conf*.

Листинг: Makefile.push

```
#
# Использование Makefile для переноса файлов конфигурации (*.conf)
# на ведомую систему.
#
```

```
SLAVE= www2.oreillynet.com
APACHE= /usr/local/apache
RM= /bin/rm
TOUCH= /bin/touch
SSH= /usr/local/bin/ssh
SCP= /usr/local/bin/scp

.SUFFIXES: .conf .ts

all: test restart sites.ts globals.ts httpd.ts

configtest: test

test:
@echo -n "Testing Apache configuration: "
@$(APACHE)/bin/apachectl configtest

restart:
$(APACHE)/bin/apachectl restart

.conf.ts:
@$(RM) -f $@
@$(SCP) $*.conf $(SLAVE):$(APACHE)/conf
@$(SSH) $(SLAVE) $(APACHE)/bin/apachectl restart
@$(TOUCH) $@
```

Поскольку реально мы не создаем никаких новых файлов, программе make несколько затруднительно решить, подвергались ли на самом деле модификации те или иные файлы настроек. Мы обманываем программу за счет создания пустых файлов *.ts* (сокращение от timestamp — временная метка), которые служат исключительно для хранения даты и времени последнего обновления. При внесении изменений в реальные файлы (*httpd.conf, sites.conf* либо *globals.conf*) нам понадобится вначале запустить команду apachectl configtest для проверки правильности настроек. Если все пройдет правильно, произойдет локальный перезапуск сервера Apache, копирование только что измененных файлов на ведомый сервер и перезапуск Apache на ведомом сервере. И, наконец, обращение к соответствующим файлам *.ts* позволяет избежать их повторной обработки, пока не будут внесены изменения в реальные файлы *.conf*. Это значительно уменьшает количество операций ввода с клавиатуры, а также ускоряет процесс по сравнению с операциями вручную при каждом обновлении.

СОВЕТ №14

Подбор нового доменного имени

(Проверка доступности имени, под которым вы хотели бы зарегистрировать свой собственный домен)

Существует множество общедоступных онлайновых утилит, предназначенных для помощи в выполнении запросов whois, которые позволяют определить, доступно ли для использования то или иное имя домена, и если нет, то кто зарегистрировал домен с таким именем. Обычно подобные утилиты помещаются непосредственно на веб-страницах, позволяют отправлять несколько запросов за один раз и часто предлагают незанятые альтернативы, если запрошенное имя уже используется.

Но если в первую очередь интерес представляет не присвоение определенного имени, а поиск его соответствия, тогда почему бы не разрешить поработать вместо вас командной строке? Предположим, необходимо найти список всех доменных имен, заканчивающихся на литеры st:

```
cat /usr/share/dict/words | grep 'st$' | sed 's/st$/.st/' | \
while read i; do \
(whois $i | grep -q '^No entries found') && echo $i; sleep 60; \
done | tee list_of_st_domains.txt
```

В итоге будет предоставлена таблица всех существующих доменных имен, заканчивающихся на st, зарегистрированных в Republica Democratica de Sao Tome e Principe (регистратор доменов для пространства имен st). В этом примере выполняется поиск всех доменных имен, соответствующих заданному критерию, по одному за раз, каждые 60 секунд. Все отсутствующие записи будут сохранены в файле под именем *list_of_st_domains.txt*, причем за ходом обработки можно наблюдать по индикатору выполнения процесса. Замените буквы st на любые другие литеры зарегистрированных TLD (Top Level DOMAIN – доменных расширений верхнего уровня), например us или to, чтобы перейти к поиску интересующего имени в другом пространстве имен.

СОВЕТ №15

Охота на пожирателей дискового пространства

(Навигация по файловой системе с помощью псевдонима в поисках виновников переполнения дискового пространства)

Как правило, это случается поздним субботним вечером. Вас вызывают, поскольку раздел одного из серверов (чаще всего в роли такового выступает почтовый сервер) практически переполнен.

Для выяснения количества свободного пространства осталось на диске, воспользуйтесь командой df:

```
rob@magic:~$ df
```

```
Filesystem 1k-blocks Used Available Use% Mounted on
/dev/sda1 7040696 1813680 4863600 27% /
/dev/sda2 17496684 13197760 3410132 79% /home
/dev/ddb1 8388608 8360723 27885 100% /var/spool/mail
```

Но и так известно, что почтовый буфер переполнен (именно поэтому вас оторвали от более приятного времяпрепровождения, никак не связанного с почтовыми серверами). Как же в кратчайшие сроки выяснить, кто является виновником данной ситуации? Ниже приводится строка, которую удобно иметь в файле вашего профиля (*.profile*):

```
alias ducks='du -cks * |sort -rn |head -11'
```

Задав этот псевдоним и запуская команду ducks в любой директории, можно получить информацию об общем объеме используемого дискового пространства, а также список десяти пользователей, занимающих наибольшее количество места на сервере, в порядке убывания. Очень удобен тот факт, что при подсчете учитываются также поддиректории (однако это может потребовать значительного количества времени на перегруженном сервере либо для папок с большим количеством вложенных директорий и файлов). Итак, посмотрим результат:

```
rob@magic:~$ cd /var/spool/mail
rob@magic:~/var/spool/mail$ ducks
8388608 total
1537216 rob
55120 phil
48800 raw
43175 hagbard
36804 mal
30439 eris
30212 ferris
26042 nick
22464 rachel
22412 valis
```

Оказывается, причиной переполнения раздела стали ваши собственные почтовые сообщения. Они занимают больше места, чем чьи-либо другие. Очевидно, что с этим нужно что-то делать, например установить новое оборудование, чтобы иметь возможность выделить больше дискового пространства под раздел */var/spool/mail.*

Поскольку данная команда позволяет выводить информацию по вложенным директориям, она отлично подойдет для периодического сбора статистических сведений по объему домашних каталогов:

```
root@magic:/home# ducks
[ спустя несколько секунд ]
```

```
13197880 total
2266480 ferris
1877064 valis
1692660 hagbard
1338992 raw
1137024 nick
1001576 rob
925620 phil
870552 shared
607740 mal
564628 eris
```

Чтобы выполнить простейшую выборочную проверку и найти виновников, приведших к отсутствию свободного дискового пространства, вызовите команду ducks, чтобы каждый раз не набирать несколько строк программного кода, хотя если вместо ducks присвоить этой команде имя ds, на клавиатуре понадобится набирать еще меньше, однако это уже нельзя будет назвать охотой на «дичь»).

СОВЕТ **№ 16**

Развлечения с /proc

(Просмотр таблицы активных процессов ядра и системных переменных)

Файловая система */proc* содержит представление таблицы активных процессов ядра. Путем нехитрых манипуляций с файлами и директориями в разделе */proc* можно узнать и даже изменить самые разные параметры рабочей системы. Однако эксперименты с */proc* могут быть чрезвычайно опасны, поскольку пользователь root обладает правами на перезапись практически любого файла в таблице процессов. Одна ошибка редиректора, и вся ОС Linux помашет вам на прощанье ручкой.

Ниже рассматривается ряд интересных действий, которые можно осуществлять с разделом */proc*. Эти примеры реализованы для последней вышедшей на момент написания книги версии ядра — 2.4; кроме того, предполагается наличие прав пользователя root. В противном случае просматривать и модифицировать процессы возможно только со своим uid. В первом случае рассмотрим слабо загруженную систему:

```
root@catlin:/proc# ls
1/ 204/ 227/ 37/ bus/ hermes/ loadavg scsi/ version
1039/ 212/ 228/ 4/ cmdline ide/ locks self@
1064/ 217/ 229/ 5/ cpuinfo interrupts meminfo slabinfo
1078/ 220/ 230/ 6/ devices iomem misc stat
194/ 222/ 231/ 698/ dma ioports modules swaps
197/ 223/ 232/ 7/ driver/ irq/ mounts sys/
2/ 224/ 233/ 826/ execdomains kcore net/ sysvipc/
200/ 225/ 254/ 827/ filesystems kmsg partitions tty/
202/ 226/ 3/ apm fs/ ksyms pci uptime
```

Директории, в качестве названий которых выступают цифровые значения, содержат информацию об активных процессах системы. Число соответствует идентификатору процесса (PID). Оставшиеся файлы и директории соответствуют драйверам, счетчикам и другим внутренним функциям работающего ядра. Интерфейс работает, как и с любыми другими файлами либо устройствами системы, путем считывания и записи каждой записи так, как если бы это был файл. Предположим, вас интересует, какая версия ядра запущена в данный момент:

```
root@catlin:/proc# cat version
Linux version 2.4.18 (root@catlin) (gcc version 2.95.3 20010315
(release))
#2 Sat Jun 22 19:01:17 PDT 2002
```

Обычно большую часть информации можно узнать, просто запустив команду uname -a, однако при этом будут отсечен «человек в середине», и выполнено то, что на самом деле делает команда uname.

Вас интересует, сколько ОЗУ установлено на вашей системе? Обратитесь к kcore:

```
root@catlin:/proc# ls -l kcore
-r-------- 1 root root 2013330688 Aug 28 21:39 kcore
```

Похоже на то, что на нашей системе установлено 192 Мб ОЗУ (2013330688/1024/1024 = 192). Кстати, вы обратили внимание на заданные файловые разрешения? Это защита системы от попыток обратиться к памяти напрямую со стороны обычных пользователей. Конечно, если вы являетесь пользователем root, то сможете делать все, что угодно. Ничто не помешает применить команду grep или strings к kcore и наблюдать за исходом. Здесь мы имеем дело с памятью системы, в которую кэшируются данные до момента перезаписи, допуская восстановление случайно потерянных данных либо отслеживание непорядочных пользователей, распоряжающихся тем, что им не принадлежит. Рыскать в kcore не слишком интересно, и поэтому данный способ используется только самыми отчаянными либо самыми скучающими людьми.

Перечислим еще некоторые достойные упоминания файлы состояния:

```
root@catlin:/proc# cat interrupts
CPU0
0: 34302408 XT-PIC timer
1: 2 XT-PIC keyboard
2: 0 XT-PIC cascade
3: 289891 XT-PIC orinoco_cs
8: 1 XT-PIC rtc
9: 13933886 XT-PIC eth0
10: 25581 XT-PIC BusLogic BT-958
14: 301982 XT-PIC ide0
```

```
NMI: 0
ERR: 0
```

Эти файлы представляют собой системные счетчики, записывающие все произошедшие прерывания, включая информацию об их номере и имени вызвавшего их драйвера.

```
root@catlin:/proc# cat partitions
major minor #blocks name

8 0 8971292 sda
8 1 8707198 sda1
8 2 257040 sda2
3 64 29316672 hdb
3 65 29310561 hdb1
```

Это список всех разделов жесткого диска и устройств, обнаруженных во время загрузки, с указанием размеров. Как видите, у нас есть диск SCSI объемом 9 Гб и IDE-устройство на 30 Гб. Здесь показаны все доступные разделы, независимо от того, смонтированы ли они на текущий момент, позволяя таким образом выяснить, присутствуют ли в системе несмонтированные диски.

Ниже представлена структура отдельного процесса:

```
root@catlin:/proc# cd 1
root@catlin:/proc/1# ls -l
total 0
-r--r--r-- 1 root root 0 Aug 28 22:05 cmdline
lrwxrwxrwx 1 root root 0 Aug 28 22:05 cwd -> //
-r-------- 1 root root 0 Aug 28 22:05 environ
lrwxrwxrwx 1 root root 0 Aug 28 22:05 exe -> /sbin/init*
dr-x------ 2 root root 0 Aug 28 22:05 fd/
-r--r--r-- 1 root root 0 Aug 28 22:05 maps
-rw------- 1 root root 0 Aug 28 22:05 mem
lrwxrwxrwx 1 root root 0 Aug 28 22:05 root -> //
-r--r--r--1 root root 0 Aug 28 22:05 stat
-r--r--r--1 root root 0 Aug 28 22:05 statm
-r--r--r--1 root root 0 Aug 28 22:05 status
```

В этой папке размещены три интересные символические ссылки: cwd указывает на текущую рабочую директорию данного процесса (можете, например, выполнить команду cd /proc/3852/cwd, чтобы перейти в директорию, из которой был запущен процесс с ID 3852), символическая ссылка exe указывает полный путь к вызываемому двоичному файлу, а root ссылается на принадлежащий процессу корневой каталог. Как правило, ссылка root практически всегда указывает на /, в противном случае происходит запуск chroot.

Файлы *cmdline* и *environ* содержат командную строку в ее оригинальном виде и полный список процессов окружения. В качестве раз-

делителя записей используются NULL-символы, поэтому, чтобы просмотреть список в удобочитаемой форме, воспользуйтесь следующей командой:

```
root@catlin:/proc/1# cat environ |tr '\o' '\n'
HOME=/
TERM=linux
BOOT_IMAGE=catlin
```

либо, в качестве более наглядного примера, обратитесь к файлу */proc/self/environ*, где отражено окружение для текущего активного процесса:

```
root@catlin:/proc/1# cat /proc/self/environ |tr '\o' '\n'
PWD=/proc/1
HOSTNAME=catlin.nocat.net
MOZILLA_HOME=/usr/lib/netscape
ignoreoff=10
LS_OPTIONS= --color=auto -F -b -T 0
MANPATH=/usr/local/man:/usr/man:/usr/x11R6/man
LESSOPEN=|lesspipe.sh %s
PS1=\u@\h:\w\$
PS2=>
...
```

и т.д. Этот прием удобно использовать в сценариях оболочки или других программах для получения дополнительных сведений об активных процессах. Применяйте его с осторожностью, не забывая о том, что непривилегированный пользователь обычно имеет доступ только к информации о запущенных им процессах.

И в завершение приведен пример практического использования файловой системы */proc*. Результаты выполнения команды ps можно сверить с таблицей активных процессов:

```
# ls -d /proc/* | grep [0-9]|wc -l; ps ax |wc -l
```

Таким образом, осуществляется небольшая выборочная проверка количества активных процессов по сравнению с числом, возвращенным командой ps. Во многих хакерских «rootkit» — наборах утилит, используемых злоумышленниками после получения первоначального доступа к системе — устанавливается взломанная версия утилиты ps, позволяющая им избирательно скрывать активные процессы, не отображая их в результатах выполнения программы ps. Но хотя можно скрыть активный процесс от программы ps, спрятать его в /proc гораздо сложнее. Если второе число намного больше первого, что произойдет, в частности, если запустить эту команду несколько раз в одной строке, нужно немедленно отключать компьютер от сети и заниматься его детальной диагностикой.

СОВЕТ №17 Оперирование символьными именами процессов в procps

(Замена идентификатора процесса PID отправкой сигналов и изменением приоритетов процессов по их названию, терминалу либо имени пользователя)

Если вы часто пользуетесь командой ps awux | grep что_то для вычисления PID процесса, который требуется уничтожить, стоит взглянуть на более современные пакеты управления процессами.

Вероятно, одним из наиболее известных пакетов утилит для работы с процессами является procps, тот же пакет, который включает в себя версию вездесущей команды top для Linux. Утилита top является настолько мощной и гибкой, что она сама по себе заслуживает отдельного рассмотрения. Более подробно прочесть о ней можно в разделе *«Мониторинг системных ресурсов при помощи утилиты top»* **[Совет № 58]**.

Среди утилит, встречающихся в пакете procps, можно назвать утилиту skill, позволяющую отправлять процессам сигналы, идентифицируя их по имени, терминалу либо идентификатору, и утилиту snice, делающую то же самое, но при этом изменяющую приоритет процессов, вместо того чтобы отправлять им сигналы.

Например, чтобы заморозить пользователя на терминале pts/2:

```
# skill -STOP pts/2
```

Для вывода пользователя из этого состояния, вам понадобится другая команда:

```
# skill -CONT pts/2
```

Чтобы изменить приоритет всех процессов пользователя luser на 5, выполните следующую команду:

```
# snice +5 luser
```

Утилита pkill напоминает по своей функциональности skill, но при этом обладает более формальными параметрами. Чтобы не заставлять команду догадываться самостоятельно о типе ссылки на процесс, при помощи имени пользователя, имени процесса либо терминала, можно задать это при помощи переключателей. Например, результаты выполнения двух следующих команд будут абсолютно одинаковыми:

```
# skill -KILL rob bash
# pkill -KILL -u rob bash
```

разве что для запуска команды pkill понадобится набрать на клавиатуре больше символов, зато она действует недвусмысленно и неважно, если имя пользователя и название процесса совпадают.

Утилита pgrep функционирует подобно pkill, но вместо того чтобы отсылать сигналы каждому процессу, она выводит соответствующие идентификаторы процессов (PID) в STDOUT:

```
$ pgrep httpd
3211
3212
3213
3214
3215
3216
```

И, наконец, команда vmstat в удобном виде предоставляет информацию о виртуальной памяти и статистику процессора:

```
$ vmstat
procs memory swap io system cpu
r b w swpd free buff cache si so bi bo in cs us sy id
0 0 0 5676 6716 35804 58940 0 0 9 9 7 9 0 0 29
```

Если необходимо постоянно следить за изменениями, задайте периодичность обновления информации в командной строке. Это число определит задержку в секундах, по истечении которой на экран будут выводиться новые результаты измерений.

Изучение малоизвестных утилит, входящих в состав пакета procps, позволит сэкономить массу времени для набора команд, не говоря уже об излишних метаниях. Воспользуйтесь командами skill либо pkill, чтобы избежать неправильного ввода аргумента PID в команде kill, приводящего к внезапному отключению службы sshd (к ярости многих пользователей, направленной на ваши хрупкие плечи системного администратора).

Дополнительная информация:

- *ftp://people.redhat.com/johnsonm/procps/;*
- *«Мониторинг системных ресурсов при помощи команды top»* **[Совет № 58]**;
- страницы оперативного руководства (man) для утилит pkill, pgrep, skill, snice и vmstat.

СОВЕТ №18 Распределение системных ресурсов между отдельными процессами

(Предотвращение захвата всех системных ресурсов пользовательскими процессами)

Случайно или преднамеренно, но один-единственный пользователь в состоянии захватить все системные ресурсы, приводя к снижению

быстродействия системы либо ее полному краху. Один из способов борьбы с пожирателями системных ресурсов, о котором часто забывают, заключается в использовании функциональности команды ulimit оболочки командной строки bash.

Чтобы не дать родительскому (либо дочернему) процессу создавать файлы слишком большого размера, воспользуйтесь командой ulimit -f (задав максимально допустимый размер файла в килобайтах):

```
rob@catlin:/tmp$ ulimit -f 100
rob@catlin:/tmp$ yes 'Spam spam spam spam SPAM!' >spam.txt
File size limit exeeded
rob@catlin:/tmp$ ls -l spam.txt
-rw-r--r--1 rob users 102400 Sep 4 17:05 spam.txt
rob@catlin:/tmp$
```

Другие пользователи могут даже уменьшить для себя этот лимит, но не увеличивать его выше установленного значения (как при помощи функций nice и renice). Это означает, что ограничение, заданное в файле */etc/profile*, в дальнейшем не может быть увеличено ни одним пользователем, кроме пользователя root:

```
rob@catlin:/tmp$ ulimit -f unlimited
bash: ulimit: cannot modify limit: Operation not permitted
```

Обратите внимание: нельзя запретить пользователю создавать множество мелких файлов, суммарный объем которых может превысить ограничение. Пользователи, замеченные в склонности к подобным действиям, должны быть проинструктированы при помощи утилиты LART (Luser Attitude Readjustment Tool — средство исправления отношения пользователей Linux). Либо, в качестве альтернативы, можете ознакомиться с дисковыми квотами, хотя их реализацию развлечением не назовешь, поэтому обычно легче исправить проблему вышеупомянутым способом. С помощью команды ulimit можно также ограничить максимальное количество дочерних процессов, которые разрешено порождать отдельному пользователю:

```
rob@catlin:~$ cat > lots-o-procs
#!/bin/bash
export RUN=$((RUN + 1))
echo $RUN...
$0
^D
rob@catlin:~$ ulimit -u 10
rob@catlin:~$ ./lots-o-procs
1...
2...
3...
```

```
4...
5...
6...
7...
8...
9...
./lots-o-procs: fork: Resource temporarily unavailable
rob@catlin:~$
```

Благодаря этой команде можно ограничить максимально допустимое количество процессов, запущенных одним пользователем на всех терминалах, включая фоновые процессы. Причем сделать это можно только таким способом, поскольку как только в процессе произойдет ветвление, связь с управляющим терминалом будет потеряна. (А как потом подсчитать количество процессов для вложенной оболочки?)

Команда ulimit имеет еще один полезный флажок: -v, определяющий максимальный объем виртуальной памяти. Как только пользователь достигнет заданного потолка, процесс будет завершен с сообщением об ошибке сегментации (что, конечно, нельзя назвать идеальным решением, но таким образом можно предотвратить крах системы в целом из-за недостатка ОЗУ либо места для файла подкачки). Если вы имеете дело с некорректно ведущим себя процессом, который пытается захватить все доступные ресурсы, например, сервер Apache + mod_perl + безграмотно написанный сценарий CGI, можно настроить ulimit на работу в качестве «безопасного тормоза» на время отладки реального источника проблемы. Задавать ограничение следует в килобайтах.

Чтобы просмотреть настройки команды ulimit, воспользуйтесь флажком -a:

```
rob@catlin:~$ ulimit -a
core file size (blocks) 0
data seg size (kbytes) unlimited
file size (blocks) unlimited
max locked memory (kbytes) unlimited
max memory size (kbytes) unlimited
open files 1024
pipe size (512 bytes) 8
stack size (kbytes) 8192
cpu time (seconds) unlimited
max user processes 1536
virtual memory (kbytes) unlimited
```

Из примера видно, что без установки ограничений на уровне системы пользовательские процессы могут достигать огромных размеров. В tcsh аналогом вышеупомянутой команды является limit:

```
rob@catlin:~> limit
cputime unlimited
```

```
filesize unlimited
datasize unlimited
stacksize 8192 kbytes
coredumpsize 0 kbytes
memoryuse unlimited
descriptors 1024
memorylocked unlimited
maxproc 1536
openfiles 1024
```

Установка ограничений на ресурсы может показаться драконовской мерой, но на самом деле это лучшая альтернатива впаданию пользовательских процессов в сумеречное состояние.

Дополнительная информация:

- *http://www.tuxedo.org/~esr/jargon/html/entry/LART.html*

СОВЕТ №19 Удаление «хвостов» за бывшими пользователями

(Убедитесь, что вы закрыли все входы и выходы для бывших пользователей)

Случается и так. Планы меняются, акценты компании смещаются, и на место одних людей приходят другие. На каком-то этапе любому администратору приходится удалять права доступа к оболочке для людей, покинувших компанию в хорошем или плохом настроении.

Когда возникает необходимость заблокировать старые учетные записи, лучше всего мыслить стратегически. Не следует рассчитывать, что только благодаря запуску команды passwd -l данный пользователь не сможет получить повторного доступа к системе. Предположим, необходимо заблокировать бывшего пользователя под именем luser. Для этого придется проверить несколько очевидных (и ряд не столь очевидных) моментов в процессе удаления «хвостов» за ним:

```
passwd -l luser
```

Разумеется, в первую очередь необходимо заблокировать пользовательский пароль.

```
chsh -s /bin/true luser
```

Еще один популярный способ заключается в смене оболочки командной строки данного пользователя на нечто, существующее только в данный момент времени. Обычно таким образом можно предотвратить доступ пользователя к серверу при помощи оболочки командной строки. Но будьте осторожны, если на системе запущена утилита sshd и разрешена удаленная аутентификация пользователей при помощи

ключей RSA и DSA: тогда пользователь luser сможет переназначать порты на любом компьютере, находящемся в зоне досягаемости сервера! При помощи следующей команды:

```
luser@evil:~$ ssh -f -N -L8000:private.intranet.server.com:80 old.server.com
```

пользователь luser переназначит локальный порт 8000 на внутренний http-порт сервера интранета. Это возможно благодаря тому, что в данном случае вместо пароля используется ключ RSA, а пользователь не собирается запускать программы на old.server.com (для этого был задан ключ -N).

Очевидно, что необходимо удалить *~luser/.ssh/authorized_keys**, чтобы в первую очередь не позволить пользователю luser воспользоваться ключом ssh. Аналогичным образом следует поступить с файлами:

```
~luser/.shosts
~luser/.rhosts
```

Обычно это не проблема, если только не запущена служба rsh либо эти функции были включены в ssh. Но как узнать, не решит ли следующий администратор включить функции *.shosts* либо *.rhosts*? Поэтому разумнее удалить все имеющиеся файлы сейчас, если такие существуют.

Обладает ли пользователь luser правами на запуск утилиты sudo? На всякий случай проверим это при помощи команды visudo.

А как насчет заданий, назначенных в планировщике cron или at?

```
crontab -u -luser -e
atq
```

Проверим, не выполняются ли какие-нибудь задания в текущий момент времени:

```
ps awux |grep -i ^luser
```

либо, как описано в разделе *«Оперирование символьными именами процессов в procps»* **[Совет № 17]**, воспользуйтесь командой:

```
skill -KILL luser
```

Проверим, в состоянии ли пользователь luser запускать сценарии cgi со своей домашней директории (или откуда-то еще):

```
find ~luser/public_html/ -perm +111
```

А что насчет PHP и других встроенных языков сценариев?

```
find ~luser ~public_html/ -name '*.php*'
```

Имеются ли у пользователя luser какие-нибудь настройки по перенаправлению электронной почты? Такой пользователь нередко может иметь привилегии на выполнение произвольных команд.

```
less ~luser/forward
grep -C luser /etc/mail/aliases
```

И, наконец, не принадлежат ли нашему пользователю luser какие-то файлы в странных местах:

```
find /-user luser > ~root/luser-files.report
```

Одним из довольно простых и быстрых способов, гарантирующих отключение всех личных файлов настройки, является перемещение домашнего каталога пользователя — выполнение команды mv /home/luser /home/luser.removed. В результате содержимое домашнего каталога пользователя luser останется нетронутым, но при этом вам не придется беспокоиться о пропущенных точках проникновения в систему. Учтите, что при этом может нарушиться целостность файла *.forward*, а также файла *~luser/public_html* и любых других общедоступных файлов, которые могли храниться в домашней директории пользователя. Поэтому, если вы решите пойти этим путем, не забывайте о вышеупомянутой проблеме (скажем, добавив соответствующую запись в системный файл *aliases* и переместив отредактированную версию файла *public_html* обратно в папку */home/luser/public_html/*).

Следует проверить все файлы настроек для приложений, с которыми работал пользователь luser, в частности, службы, запускаемые под привилегированными пользователями. Даже безобидные файлы настроек пользователя сервера Apache могут быть использованы впоследствии для обеспечения доступа к оболочке.

Этот список является демонстрацией того, что для лишения бывшего пользователя всех прав доступа недостаточно только блокировать пользовательский пароль. Если этот пользователь когда-нибудь обладал правами пользователя root, ситуация становится еще более сомнительной. Позднее доступ к системе может быть получен с помощью троянского коня, невидимого для модуля ядра, либо путем простого изменения пароля пользователя root.

СОВЕТ №20

Удаление неиспользуемых драйверов из ядра системы

(Оптимизируйте ядро ради ускорения загрузки и долговременной стабильной работы)

ОС Linux может работать с самым разнообразным аппаратным обеспечением. В нее встроена поддержка различного оборудования, начиная с жестких дисков и ОЗУ, и заканчивая менее распространенными устройствами, например сканерами USB и платами видеозахвата. Ядро, поставляемое в составе большинства дистрибутивов Linux, является наиболее полным и безопасным за счет некоторого снижения его эффективности из-за встроенной поддержки максимального количества устройств.

Во время загрузки компьютера обратите внимание на выводимые ядром сообщения. Там можно найти сообщения о различном оборудовании типа SCSI-контроллеров и карт Ethernet, которое на самом деле не подключалось к системному блоку. Если в дистрибутиве, который имеется в вашем распоряжении, отключен вывод загрузочных сообщений ядра, воспользуйтесь для их просмотра командой dmesg (возможно, вызываемой через less).

Чтобы ускорить загрузку системы и избежать проблем несовместимости между неиспользуемыми драйверами и установленным аппаратным обеспечением, следует ограничить список загружаемых драйверов.

Существует две школы по управлению драйверами ядра. Одни предпочитают создавать ядро, встраивая в него всю необходимую функциональность, не прибегая к помощи загружаемых модулей ядра. Другие создают «облегченные» версии ядра, в которых загрузка необходимых драйверов проходит отдельно во время старта системы. Разумеется, каждому методу присущи свои достоинства и недостатки.

К примеру, монолитное ядро (без загружаемых модулей) более стабильно — оно загрузится всегда, даже в случае повреждения драйверов, хранимых в папке */lib/modules*. Некоторые администраторы доходят до того, что вообще отключают поддержку загружаемых модулей ядра, чтобы исключить возможность установки в системе троянских приложений, замаскированных случайным злоумышленником под какой-нибудь драйвер. С другой стороны, в случае подключения к компьютеру дополнительного оборудования придется полностью перекомпилировать ядро.

Использование в системе загружаемых модулей ядра предоставляет гибкость загрузки драйверов ядра — можно менять порядок их загрузки и даже передавать им различные параметры без перезагрузки. Недостатком такого подхода является необходимость наличия рабочих копий модулей в папке */lib/modules*; в противном случае, при возникновении проблем система окажется не в состоянии загрузить драйверы. Выполняя компиляцию нового ядра, не забывайте запускать команду make modules_install и своевременно обновлять утилиты модулей ядра, такие как modprobe и lsmod. Построение ядра с загружаемыми модулями также подойдет, если монолитное ядро слишком велико для загрузки, т.к. запуск дополнительных драйверов устройств произойдет после загрузки самого ядра и установки файловой системы.

Независимо от выбранного метода необходимо создать ядро, обладающее минимально необходимой функциональностью для загрузки системы. То есть, оно должно включать драйверы для устройств IDE, SCSI и другого оборудования (возможно, дисковода гибких дисков или даже сетевой карты), с которого будет производиться загрузка. Необходима также поддержка файловой системы корневого раздела диска (ext2, ext3 либо reiserfs). Убедитесь, что скомпилирована версия ядра

системы, поддерживающая то количество памяти, которое имеется на вашей системе (см. раздел *«Использование больших объемов ОЗУ»* **[Совет № 21]**). Выберите процессор, соответствующий установленному оборудованию, чтобы убедиться, что на системе включены все возможности оптимизации. Не забудьте скомпилировать ядро SMP, поддерживающее симметричную многопроцессорную обработку, если на системе установлено больше одного процессора.

Чтобы скомпилировать собственное ядро, необходимо распаковать исходные коды ядра туда, где для этого достаточно места, например, в раздел */usr/local/src/linux*. Выполните команду make menuconfig либо, если запущена система X, команду make xconfig. Аккуратно отнеситесь к выбору драйверов (нажимая Y для добавления драйвера в список встроенных и M для загружаемых модулей). Помните, что создаете ядро для сервера, которому не нужны дополнительные устройства. К примеру, действительно ли серверу необходима поддержка звукового устройства, даже если оно встроено в материнскую плату? А как насчет USB? Если USB-устройство не используется для выполнения заданий, непосредственно связанных с функциональностью сервера, не стоит устанавливать для него драйвер. Также подумайте о возможности отключения в BIOS всего ненужного оборудования, чтобы сберечь системные ресурсы и снизить потенциальный риск возникновения конфликтов оборудования.

Сформировав список модулей ядра, которые могут быть отключены, сохраните для дальнейшего использования файл настроек под названием *.config* в корневой папке дерева каталогов с исходными кодами ядра. Это значительно облегчит обновление ядра в будущем, т.к. его можно будет скопировать на верхний уровень дерева каталогов с новыми исходными кодами ядра и запустить команду make oldconfig. Теперь скомпилируйте обновленное ядро (и модули, если это необходимо), установите их и проверьте в работе. Убедитесь в работоспособности всех устройств после установки нового программного обеспечения и внимательно проследите за всеми загрузочными сообщениями. Обратите внимание на все неожиданные предупреждения либо ошибки и постарайтесь тут же их исправить, чтобы они не привели к серьезным неприятностям в будущем. И, наконец, при удаленной работе с ядром не забудьте включить в него поддержку сетевых устройств! Нет ничего печальнее, чем обнаружить, что вы забыли встроить сетевые драйверы сразу после выполнения команды shutdown -r now.

Построение хорошо оптимизированного ядра – далеко не тривиальная задача, однако ее решение позволяет получить сервер, работающий с максимальной эффективностью. Не сдавайтесь: даже если для достижения цели понадобится предпринять несколько попыток, ваши труды окупятся сторицей.

Дополнительная информация:

- «Работа в Linux» (Running Linux), четвертое издание (O'Reilly);
- *«Использование больших объемов ОЗУ»* **[Совет № 21]**.

СОВЕТ **№21**

Использование больших объемов ОЗУ

(Заставьте систему Linux использовать всю доступную системную память)

ОС Linux способна адресовать до 64 Гб физического ОЗУ на x86-совместимых системах. Но, чтобы на самом деле иметь возможность работать с памятью свыше 960 Мб, необходимо настроить систему соответствующим образом. Если установить в компьютер больше оперативной памяти, не настроив систему, память свыше 960 Мб просто окажется недоступной. Распространенная жалоба обычно заключается в том, что сразу после установки 1 Гб вы видите сообщение системы о свободных 960 Мб, притом что во время загрузки компьютера выводилось сообщение POST о наличии 1024 Мб.

Способ адресации доступной системной памяти ядром системы определяется настройками поддержки верхней области памяти (CONFIG_NOHIGHMEM). В зависимости от объема памяти, который намерены использовать, задайте нужные настройки:

```
up to 960MB: off
up to 4GB: 4GB
more than 4GB: 64GB
```

Учтите, что при использовании 64 Мб ОЗУ в процессоре должен поддерживаться режим физического расширения адресов PAE (Physical Address Extension). Этот режим поддерживают все процессоры, начиная с Intel Pentium Pro, однако он отсутствует на более старых процессорах, в результате чего ядро откажется загружаться — по этой причине данная настройка отключена по умолчанию. Выберите, что вам нужно, и перекомпилируйте ядро, как описано в разделе *«Удаление неиспользуемых драйверов из ядра системы»* **[Совет № 20]**.

После компиляции и установки ядра может понадобиться сообщить загрузчику, сколько памяти установлено на системе, чтобы ядро знало об этом во время загрузки, поскольку не все версии BIOS правильно определяют объем установленной памяти. Для этого задайте в загрузчике параметр mem=. Предположим, на компьютере установлено 2 Гб ОЗУ. Если используется оболочка Lilo, добавьте в файл конфигурации */etc/lilo.conf* следующую строку:

```
append="mem=2048M"
```

Если используется Grub, добавьте в файл */etc/grub.conf* строку:

```
kernel /boot/vmlinuz-2.4.19 mem=2048M
```

А если у вас запущена loadlin, передайте этот параметр непосредственно в командной строке loadlin:

```
c:\loadlin c:\kernel\vmlinuz root=/dev/hda3 ro mem=2048M
```

Хотя, если вы работаете с loadlin, зачем читаете книгу «100 хакерских советов для сервера под управлением Linux»?

СОВЕТ №22

hdparm: тонкая настройка параметров IDE-устройств

(Добейтесь от своего оборудования максимальной производительности)

Если вы работаете с системой Linux, на которой установлено по крайней мере одно (E)IDE устройство, и никогда не слышали об hdparm, обязательно прочитайте этот раздел.

В большинстве дистрибутивов Linux при обращении к контроллерам и устройствам IDE ядро использует настройки по умолчанию. Они отличаются высокой консервативностью, поскольку предназначены для максимальной защиты данных любой ценой. Но, как многие со временем убеждаются, безопасность почти никогда не означает быстроту. А для приложений, обрабатывающих огромные объемы данных, никогда не существовало такого понятия, как «достаточно быстро».

Если хотите добиться от своего IDE-устройства максимальной производительности, обратите внимание на команду hdparm(8). Она позволяет не только выяснить, какие устройства работают в системе в данный момент, но и произвести их тонкую настройку.

Следует подчеркнуть, что при определенных обстоятельствах использование этих команд МОЖЕТ ВЫЗВАТЬ НЕПРЕДВИДЕННУЮ ПОРЧУ ДАННЫХ! Применяйте их на ваш риск! По крайней мере, перед тем как начать эксперименты, сделайте резервную копию данных своего компьютера и переведите его в однопользовательский режим работы.

Находясь в однопользовательском режиме, который обсуждался в разделе *«Отказ от подключения к консоли»* **[Совет № 2]**, попытаемся выяснить быстродействие нашего ведущего жесткого диска:

```
hdparm -Tt /dev/hda
```

В результате вы увидите нечто типа:

```
/dev/hda
Timing buffer-cache reads: 128 MB in 1.34 seconds = 95.52 MB/sec
Timing buffered disk reads: 64 MB in 17.86 seconds = 3.58 MB/sec
```

О чем это говорит? Параметр -T выполняет тестирование подсистемы кэширования, т.е. памяти, процессора и буфера. Параметр -t проверяет скорость считывания данных с диска без помощи кэша. Запустив оба эти теста несколько раз, получите представление о производительности подсистемы ввода/вывода. В примере приведены реальные данные, полученные в результате тестирования системы PentiumII/350/128Мб ОЗУ/EIDE HDD; ваши значения могут отличаться.

Но, даже учитывая возможный разброс значений, число в 3.58 Мб/с для вышеупомянутого оборудования просто умиляет. А в рекламе сообщается, что скорость передачи данных жестким диском может достигать 66 Мб/с!? Как же это возможно? Попробуем подробнее разобраться в том, как ОС Linux работает с жестким диском:

```
/dev/hda:
multcount = 0 (off)
I/O support = default 16-bit)
unmackirq = 0 (off)
using_dma = 0 (off)
keepsettings = 0 (off)
noerr = 0 (off)
readonly = 0 (off)
readahead = 8 (on)
geometry = 1870/255/63, sectors = 30043440, start = 0
```

Это настройки жесткого диска по умолчанию. Самые безопасные, но не самые оптимальные. А как насчет 16-битного режима? Разве он не канул в лету вместе с эпохой 386!?

Дело в том, что использование подобных настроек гарантирует работоспособность системы практически на любом оборудовании, с которым можно столкнуться. Но поскольку в нашем распоряжении есть нечто большее, чем запыленная 16-битная карта ввода-вывода, выпущенная 8 лет назад, поговорим о более интересных опциях:

multcount

Представляет собой счетчик секторов. Этот параметр контролирует количество считываемых с диска за одно прерывание ввода/вывода секторов. Практически все современные IDE-устройства поддерживают эту функцию. В оперативном руководстве сказано, что включение данной опции обычно снижает нагрузку на операционную систему во время использования дисковой подсистемы ввода/вывода на 30-50%. На многих системах включение данной опции также позволяет повысить скорость передачи данных на 5-50%.

I/O support

Этот флажок контролирует способ передачи данных с шины PCI на контроллер. Практически все современные контроллеры поддерживают режим 3 либо 32-битный режим w/sync. Некоторые контроллеры способны работать даже в 32-битном асинхронном режиме. Включение данной опции в большинстве случаев позволяет удвоить производительность системы (см. дальше).

unmaskirq

Включение данного параметра позволяет ОС Linux демаскировать другие прерывания во время обработки дискового прерывания. Что это означает? Дело в том, что операционная система может обрабатывать другие события, связанные с прерываниями (например, сетевой трафик), ожидая возврата запрошенных данных вашим устройством. Так можно повысить общую производительность системы, однако будьте осторожны: не всякое оборудование поддерживает этот режим. Обращайтесь к оперативному руководству.

using_dma

Использование режима DMA связано с некоторыми сложностями. Если сможете заставить свой контроллер и жесткий диск работать в режиме DMA — дерзайте. За более подробными сведениями следует обратиться к оперативному руководству.

Проверим на практике некоторые настройки, предназначенные для ускорения работы:

```
# hdparm -c3 -m16 /dev/hda

/dev/hda
setting 32-bit I/O support flag to 3
setting multcount to 16
multcount = 16 (on)
I/O support = 3 (32-bit w/sync)
```

Отлично! 32-битная поддержка – звучит вдохновляюще. Кроме того, теперь у нас работает функция упреждающего чтения. Запустим тест на производительность повторно:

```
# hdparm -tT /dev/hda

/dev/hda
Timing buffer-cache reads: 128 MB in 1.41 seconds = 90.78 MB/sec
Timing buffered disk reads: 64 MB in 9.84 seconds = 6.50 MB/sec
```

Почти удвоенная скорость без особых усилий! Невероятно.

Но минуточку: ведь мы еще не применили демаскирование прерываний, режим DMA либо хотя бы подходящий режим PIO! Конечно, включение этих режимов связано с определенным риском. На страницах оперативного руководства упоминается использование режима Multiword DMA 2, поэтому воспользуемся именно им:

```
# hdparm -X34 -d1 -u1 /dev/hda
```

К сожалению, как оказалось, этот режим не поддерживается на данной системе (компьютер под управлением Linux завис, как это случается с компьютером под управлением NT при запуске Java-приложений). Так что после загрузки (снова в однопользовательском режиме) были подобраны такие настройки:

```
# hdparm -X66 -d1 -u1 -m16 -c3 /dev/hda
/dev/hda
setting 32-bit I/O support flag to 3
setting multcount to 16
setting umaskirq to 1 (on)
setting using_dma to 1 (on)
setting xfermode to 66 (UltraDMA mode2)
multcount = 16 (on)
I/O support = 3 (32-bit w/sync)
unmaskirq = 1 (on)
using_dma = 1 (on)
```

Проверим результирующее быстродействие:

```
# hdparm -tT /dev/hda
/dev/hda
Timing buffer-cache reads: 128 MB in 1.43 seconds = 89.51 MB/sec
Timing buffered disk reads: 64 MB in 3.18 seconds = 20.13 MB/sec
```

20.13 Мб/с. Разительное отличие от смехотворных 3.58 Мб/с, с которых мы начали.

Обратите внимание на повторное задание переключателей -m16 и -c3. Дело в том, что после перезагрузки, теряются настройки, заданные в hdparm. Убедившись в стабильности работы системы с вышеупомянутыми настройками, добавьте эту строку в сценарий */etc/rc.d/**. Делать это желательно после запуска команды fsck, поскольку расширенная проверка файловой системы с помощью контроллера в неустойчивом режиме — отличный способ генерирования огромных количеств энтропии, но не лучший метод администрирования системы. Если вы не можете найти на своей системе команду hdparm, которая обычно размещена в каталоге */sbin* либо */usr/sbin*, загрузите ее исходные коды с сайта *http://metalab.unc.edu/pub/Linux/system/hardware/*.

Контроль версий

Советы № 23-36

Если вы серьезно подходите к администрированию, то, безусловно, тратите много времени, пытаясь задать тонкие настройки системы, чтобы добиться от нее адекватного поведения. К сожалению, не всегда можно определить, когда то или иное приложение работает правильно, а когда его работоспособность уже нарушена. Чаще всего встречаются промежуточные градации работоспособности приложения. Даже внесение незначительных изменений в конфигурацию может привести к незаметным вначале последствиям, которые не проявляют себя в течение нескольких дней или даже недель.

В этом случае могущественным союзником станут пакеты контроля версий. Помимо выполнения простых функций резервного копирования любой хороший пакет умеет отслеживать внесение изменений в файлы, например, когда они были изменены, кто автор изменений и почему эти изменения были сделаны. Такая информация имеет огромную ценность при работе со сложными системами, особенно если за ее обслуживание несут ответственность несколько человек. Сохранив критичные системные файлы в RCS (Revision Control System – простая система контроля версий) или CVS (Concurrent Versioning System – система контроля параллельных версий), всегда можно выполнить «откат» к любой предшествующей версии файла, а также проанализировать различия между его предыдущей и настоящей версиями.

Целью данной главы является рассмотрение специального синтаксиса, необходимого для работы с системами RCS или CVS, непосредственно на примерах их использования. Если пакет RCS удобен для обслуживания файлов на одиночной системе, то CVS предоставляет расширенные возможности по архивации и слиянию, поддерживая центральное хранилище (репозиторий) где-либо в сети. Понятно, что пакет CVS представляет собой более сложное средство, чем система RCS. В хакерских советах № 23-25 изложен ускоренный курс по системе RCS, а в оставшихся советах № 26-36 более подробно изложена ин-

формация о системе CVS, начиная с простейших команд и заканчивая настройкой вашего собственного анонимного репозитория CVS.

Естественно, материал, изложенный в данной главе, предполагает наличие навыков работы с файлами из командной строки, даже если вам приходилось работать с системами RCS или CVS ранее. Для получения более подробной информации о системе CVS обратите внимание на великолепную книгу «Разработка открытых кодов с помощью CVS» (Open source development with CVS) Карла Фогеля (издательство CoriolisOpen Press).

СОВЕТ №23

Начинаем работу с системой RCS

(Используйте систему RCS для управления системными файлами и сохранения истории вносимых изменений)

RCS представляет собой систему контроля версий, предназначенную для хранения нескольких наборов файлов конфигурации, сценариев оболочки и любых других текстовых файлов. В отличие от системы CVS, в RCS отсутствуют функции удаленного управления — вся информация хранится в локальной файловой системе.

Система RCS сохраняет все версии в директории *RCS/*, являющейся вложенной по отношению к текущей папке. Чтобы инициировать создание нового репозитория RCS, достаточно создать новую папку:

```
root@catlin:/etc# mkdir RCS
```

Прежде чем приступить к работе с системой, надо инициализировать файл в новой директории RCS. Возьмем, к примеру, файл *syslog.conf*:

```
root@catlin:/etc# ci -i syslog.conf
RCS/syslog.conf, v <-- syslog.conf
enter description, terminated with single '.' or end of file:
NOTE: This is NOT the log message!
>> Here's the syslog.conf for Catlin
initial revision: 1.1
done
```

Первоначальная команда ci -i означает «check and initialize – проверить и инициализировать», то есть предоставить системе RCS первую рабочую копию файла. Система запрашивает первоначальное описание, а затем перемещает файл из текущей директории. Вряд ли вы захотите сохранить такое положение вещей на длительный период времени, поэтому после начальной инициализации файла в системе RCS следует выполнить его проверку:

```
root@catlin:/etc# co syslog.conf
RCS/syslog.conf, v —> syslog.conf
```

```
revision 1.1
done
```

После этого файл будет скопирован обратно в текущую директорию. Однако мы еще не готовы приступить к работе с ним. Во-первых, выполните проверку, заблокировав данный файл, чтобы запретить другим пользователям вносить изменения, пока вы с ним работаете.

```
root@catlin:/etc# co -l syslog.conf
RCS/syslog.conf, v --> syslog.conf
revision 1.1 (locked)
done
```

Теперь можно заняться редактированием содержимого файла. После окончания проверьте файл еще раз, предварительно разблокировав при помощи команды ci -u:

```
root@catlin:/etc# ci -u syslog.conf
syslog.conf, v <-- syslog.conf
new revision: 1.3; previous revision: 1.2
enter log message, terminated with single '.' or end of file:
>> Added remote logging to the new security log host
>> .
done
```

Советуем вносить в журнал достаточно информативные записи, т.к. через полгода, если программа начнет вытворять странные вещи, вы уже не вспомните, что подразумевали под строкой «внес пару изменений».

Если возникают трудности в проверке тех или иных файлов, возможно, на каком-то этапе вы забыли ввести переключатель -l (в команде co) или -u (в команде ci). В таком случае можно начать все сначала, создав резервную копию, а затем выполнить проверку и обратное копирование, предполагая, что файл *syslog.conf* был только что отредактирован:

```
root@catlin:/etc# ci -u syslog.conf
syslog.conf, v <-- syslog.conf
ci: syslog.conf, v: no lock set by root
root@catlin:/etc#
```

Ну и ну! Лучше сделать резервную копию и начать все сначала:

```
root@catlin:/etc# cp syslog.conf syslog.conf.backup
root@catlin:/etc# co -l syslog.conf
syslog.conf, v --> syslog.conf
revision 1.4 (locked)
done
root@catlin:/etc# cp syslog.conf syslog.conf.backup
root@catlin:/etc# ci -u syslog.conf
syslog.conf, v <-- syslog.conf
new revision: 1.5; previous revision: 1.4
```

```
enter log message, terminated with single '.' or end of file:
>> commented out the FIFO line
>> .
done
```

Несколько еще более интересных моментов, связанных с работой системы RCS, содержится в следующих разделах.

СОВЕТ №24 Проверка предыдущих версий в системе RCS (Обеспечьте собственную безопасность при помощи ревизий RCS)

Рано или поздно система может прийти в такое состояние, что будет невозможно восстановить ее работоспособность. Предположим, вы только что внесли огромное количество правок в файл *httpd.conf*, после чего получили в ответ сообщение об ошибке, способное вызвать ужас в душе любого администратора:

```
root@catlin:/usr/local/apache/conf# ../bin/apachectl restart
apachectl restart: httpd not running, trying to start
apachectl restart: httpd could not be started
/* перезапуск команды apachectl: невозможно запустить httpd
```

О боже! Что вы наделали? Если у вас не будет работать веб-сервер, люди даже не получат дружественное сообщение об ошибке 404, а просто увидят сообщение о том, что подключение невозможно (Connection Refused). Останется только копаться в файле *httpd.conf* при помощи утилиты vi, чтобы выяснить, что именно вы или предыдущий администратор натворили, чтобы заслужить такую кару.

Однако, если использовать систему контроля версий, ситуация не настолько безрадостная. Чтобы увидеть различия между последней проверенной версией файла и версией с внесенными изменениями, воспользуйтесь командой rcsdiff:

```
root@catlin:/usr/local/apache/conf# rcsdiff httpd.conf
===================================================================
RCS file: RCS/httpd.conf, v
retrieving revision 1.1
diff -r1.1 httpd.conf
458c458
< ErrorLog /usr/local/apache/logs/error_log
---
> ErrorLog :wq/usr/local/apache/logs/error_log
```

Вот в чем причина! Как видите, в строке 458 были случайно вставлены символы :wq. Мы, должно быть, упустили ESC-ключ. Исправим ошибку, перезапустим сервер Apache и снова выполним проверку.

Но что, если последствия внесенных изменений проявятся не сразу? Предположим, файл настроек модифицирован во вторник утром, а проблемы начали появляться только в среду к середине дня. Как вернуться к проверенной работоспособной копии конфигурационного файла?

Очень просто: проверьте текущую версию файла при помощи переключателя -r:

```
root@catlin:/etc# co -l -r1.2 syslog.conf
syslog.conf, v --> syslog.conf
revision 1.2
done
```

Вот мы и вернулись к работоспособному файлу версии 1.2. Помните, что в системе RCS следует сохранять версии файлов как можно чаще, а лучше — после каждого внесения изменений.

СОВЕТ №25

Отслеживание изменений при помощи журнала rcs2log

(Определите с первого взгляда, кто и когда редактировал ваши файлы)

Система RCS демонстрирует свои по-настоящему мощные возможности в том случае, когда к редактированию файлов причастны несколько человек. Наличие нескольких администраторов иногда приводит к «игре в виноватого». В этой ситуации очевидно, что проблема не может возникнуть на пустом месте: как правило, она бывает вызвана внесением некоторых изменений, в которых никто не хочет сознаваться. Имея в своем распоряжении журнал системы RCS, можно легко определить, кто и какие изменения вносил за последнее время:

```
root@www:/usr/local/apache/htdocs# rcs2log index.html
2002-08-14 rob <rob@mouse>

*index.html: meeting announcement

2002-07-30 rob <rob@mouse>

*index.html: gre.tunnel announcement

2002-07-12 sderle <sderle@mouse>

*index.html: added Marin and shuffled around a bit.

2002-07-12 rob <rob@mouse>

*index.html: meeting announcement + v0.81

2002-07-01 jim <jim@mouse>
```

```
*index.html: *** empty log message ***

2002-07-01 rob <rob@mouse>

*index.html: meeting reminder
```

Так, а что в журнале делает пустая строка? Наверное, стоит поговорить с Джимом и запустить команду rcsdiff, чтобы сравнить версии файлов до и после внесенных им изменений. Но как определить, какие записи в журнале какой версии файла соответствуют? Попробуйте воспользоваться переключателем -v для журнала rcs2log:

```
root@www:/usr/local/apache/htdocs# rcs2log index.html
2002-08-14 rob <rob@mouse>

*index.html 1.54: meeting announcement

2002-07-30 rob <rob@mouse>

*index.html 1.53: gre.tunnel announcement

2002-07-12 sderle <sderle@mouse>

*index.html 1.52: added Marin and shuffled around a bit.

2002-07-12 rob <rob@mouse>

*index.html 1.51: meeting announcement + v0.81

2002-07-01 jim <jim@mouse>

*index.html 1.50: *** empty log message ***

2002-07-01 rob <rob@mouse>

*index.html 1.49: meeting reminder
```

Теперь применим команду diff для сравнения двух интересующих нас версий файла:

```
root@catlin:/usr/local/apache/conf# rcsdiff –r1.49 -r1.50 index.html
===================================================================
RCS file: RCS/index.html, v
retrieving revision 1.49
retrieving revision 1.50
diff -r1.49 -r1.50
199a200,202
> <dt><a href="http://labs.google.com/">Google Labs</a></dt>
> <dd>Take a look at the strange and wondrous things that Goggle is up to...
</dd>
>
```

Итак, речь шла всего лишь о добавлении ссылки. Конечно, гораздо проще нажать ^D при выводе запроса, однако занесение в журнал пустой строки комментариев считается дурным тоном.

Т.о., простые команды ci, co, rcsdiff и rcs2log являются очень гибким средством контроля версий. Используйте его регулярно, и, возможно, в один прекрасный день оно спасет ваши данные.

СОВЕТ №26

Начинаем работу с системой CVS

(Подробно о системе контроля параллельных версий)

Система контроля параллельных версий (CVS – Concurrent Versioning System) предназначена для управления параллельной работой файлов. Она часто применяется в ходе разработки больших программных проектов, а также может быть полезной для системных администраторов, технических авторов и других сотрудников, отвечающих за управление файлами.

Система CVS сохраняет все файлы в центральном репозитории. Они доступны для всех пользователей файлов в соответствии со стандартными разрешениями Unix. В системе существуют команды проверки копий файлов и фиксации изменений в репозитории. Кроме того, система позволяет сканировать файлы во время их перемещения в репозиторий и обратно, чтобы не допустить перезаписи одним пользователем изменений, внесенных другим пользователем. Система обеспечивает сохранение истории файлов, что весьма полезно, если ваш начальник склонен требовать восстановления функциональности, которую он считал ненужной пару месяцев назад. Вдобавок система обеспечивает резервное копирование всего проекта исключительно путем создания резервной копии всех необходимых файлов в репозитории.

Обычные пользователи

Система CVS предназначена для отдельных разработчиков либо целых групп. Отдельным разработчикам система предоставляет возможность работы на дому, из офиса либо на территории клиента без необходимости носить с собой диски с информацией. В ней также существуют функции контроля версий, позволяющие осуществлять откат без потери данных. В случае с группами разработчиков также сохраняется информация о том, кто и какие изменения вносил в тот или иной файл, предотвращая перезапись результатов работы других разработчиков.

Системные администраторы могут использовать CVS для хранения файлов конфигурации. Вы можете внести изменения, запустить коман-

ду cvs commit и протестировать результаты. А в случае возникновения проблем — выполнить откат изменений, даже если последствия проявятся через полгода после их внесения.

Администраторы могут сохранять дерево изменений конфигурации даже для целых групп серверов. Нужно добавить новый сервер? Достаточно воспользоваться cvs, чтобы проверить дерево настроек для данного типа сервера. Фиксация внесенных изменений также помогает отслеживать, кто, когда и какие изменения вносил.

В этом разделе мы поговорим о том, как запустить систему CVS и заставить ее работать быстро в качестве клиента и в роли сервера.

Создание репозитория

Репозиторий должен размещаться на машине с достаточным объемом дискового пространства, поскольку в нем будут храниться все ваши данные и вносимые изменения. Результаты, полученные эмпирическим путем, свидетельствуют о том, что репозиторий нужно сохранять на разделе, объем которого втрое превышает ожидаемый объем окончательного проекта. Затем воспользуйтесь «принципом Скотти» и удвойте результаты своей оценки — ведь проект имеет свойство получаться больше ожидаемого. Если требуется хранить и исполняемые файлы, расчеты нужно умножить на 10. Более точно оценивать объем дискового пространства, который необходимо отвести под репозиторий, вы научитесь в ходе создания своего первого проекта.

Во-первых, убедитесь, что все пользователи системы CVS располагают действующими учетными записями и имеют доступ к системе, на которой размещен репозиторий, со всех компьютеров, которые они намереваются использовать для работы.

Затем создайте в репозитории корневой каталог. Часто репозитории размещаются в папках */home/cvsroot* или */usr/local/cvsroot*. Воспользуйтесь командой cvs init для указания директории хранилища.

```
cvs -d /home/cvsroot init
```

В Debian Linux имеется сценарий под названием cvs-makerepos, создающий хранилище на основе предыдущих сценариев конфигурации Debian. За более подробной информацией обращайтесь к оперативному руководству по командам cvs-makerepos и cvsconfig для автоматизированной системы настройки репозитория Debian CVS.

В большинстве случаев на серверах CVS используется один репозиторий с несколькими модулями. Т.е. репозиторий может находиться в папке */home/cvsroot*, но при этом содержать несколько проектов (например, tools, email, dns, myproj и т.д.).

Импортирование нового модуля

Перед загрузкой проекта в систему CVS необходимо проанализировать его структуру. Перемещение либо переименование файлов может нарушить записи CVS об истории файлов. Удаление папки может привести к потере записей обо всех файлах и папках, являющихся вложенными по отношению к данной. По этой причине в системе CVS отсутствуют функции по перемещению и переименованию файлов и директорий.

Создайте начальную структуру каталогов — даже если у вас имеется всего одна директория. Инициализируйте все нужные файлы. Запустите, находясь в корневом каталоге своего проекта, команду

```
cvs -d путь_к_репозиторию import имя_модуля тег_поставщика тег_релиза.
```

В большинстве случаев информация о поставщике либо релизе проекта не понадобится. Хотя система CVS считает эти параметры обязательными, в качестве поставщика можно задать имя модуля, а в теге релиза – номер текущей версии:

```
/home/jenn$ cd example
/home/jenn/examples$ cvs -d /home/cvsroot import example example_project ver_0-1
```

Переменные среды

Если вы работаете с единственным репозиторием, укажите полный путь к нему в переменной среды $CVSROOT:

```
export CVSROOT=/home/cvsroot/
```

Настроив ее, можно в дальнейшем опускать в команде CVS параметры -d путь_к_репозиторию. Если понадобится временно работать с другим репозиторием, задайте новое значение в переменной $CVSROOT либо воспользуйтесь флажком -d для перегрузки текущих настроек $CVSROOT.

Если репозиторий CVS размещен на другом компьютере, необходимо определить, как система CVS будет получать к нему доступ. Для работы с удаленным репозиторием переменную $CVSROOT нужно настроить следующим образом :метод:имя_пользователя@хост:путь. Ее значение должно будет выглядеть приблизительно как :ext:jenn@cvs.example.com.au:/home/cvs.

По умолчанию для доступа к репозиторию, размещенному на удаленном компьютере, пользователи CVS применяют команду rsh. Это был довольно оправданный выбор несколько лет назад, но теперь запуск этой команды на рабочем сервере можно расценить как очень плохую идею. К счастью, заданный нами метод доступа ext заставляет систему CVS обратиться к переменной среды $CVS_RSH и запустить эту программу для установки подключения к удаленному компьютеру, на котором размещен репозиторий. Многие пользователи задают в пе-

ременной $CVS_RSH команду ssh, используя при подключении клиентские ключи ssh (см. раздел *«Быстрый вход в систему при помощи клиентских ключей ssh»* **[Совет № 66]**). Это значительно упрощает работу с репозиторием, поскольку не требуется набирать пароль каждый раз при выполнении обновления.

Еще одним распространенным способом доступа является команда pserver, которая в настоящее время применяется в основном для организации анонимного доступа к открытым репозиториям CVS. О том, как работать с анонимным репозиторием CVS и как настроить собственный открытый репозиторий, можно прочесть в разделе *«Быстрый вход в систему при помощи клиентских ключей ssh»* **[Совет № 66]**.

Дополнительная информация:

- «CVS в Nutshell» (O'Reilly);
- «Анонимный CVS»;
- *«Быстрый вход в систему при помощи клиентских ключей ssh»* **[Совет № 66]**.

СОВЕТ №27 CVS: проверка модуля

(Как составить работающую копию модуля CVS)

Система CVS хранит все файлы в центральном репозитории, но сами пользователи имеют дело с рабочими копиями файлов.

Создайте папку, в которой намерены хранить результаты своей работы, например *~/cvs*, затем перейдите в эту директорию. Для проверки модуля необходимо использовать следующий синтаксис команды: cvs checkout имя_модуля. Чтобы проверить модуль под названием example, команда должна выглядеть следующим образом: cvs checkout example. Команда checkout осуществляет копирование всех файлов модуля и вложенных директорий в папку *cvs*.

```
cvs$ ls
example
cvs$ cd example; ls
CVS src
cvs/example$ cd CVS; ls
Entries Repository Root
```

Эта папка используется системой CVS для своих нужд. Параметр Entries содержит список поддиректорий, о которых известно системе CVS. Параметр Repository хранит путь к соответствующим папкам в репозитории. Параметр Root, в свою очередь, содержит полный путь к репозиторию, чтобы задавать для каждого файла опцию -d путь_к_репозиторию.

Учтите, что переменная CVS/Root перегружает значение переменной среды $CVSROOT, так что, если изменится местоположение репозитория, понадобится повторно проверить модуль. В качестве альтернативы, в процессе редактирования крупного файла, обнаружив в этот момент необходимость изменения местоположения хранилища, попробуйте выполнить следующую строку на языке Perl, как показано в разделе *«Глобальный поиск и замена при помощи языка Perl»* **[Совет № 73]**:

```
cvs/src$ ls
CVS Makefile sample.h sample.c
```

В директории *src* содержатся исходные коды нашего проекта example. *sample.h*, *sample.c* и *Makefile* представляют собой рабочие копии обычных файлов. Внутри репозитория они хранятся в формате, позволяющем отслеживать все вносимые изменения.

СОВЕТ №28

Обновление рабочей копии

(Сохраните последние изменения в своем модуле CVS)

Каждый день, перед тем как приступить к работе, и каждый раз, когда кто-то другой вносит и сохраняет изменения, необходимо перейти с помощью команды cd в вашу рабочую директорию и выполнить команду cvs update. В результате будет произведено сравнение рабочих копий ваших файлов с файлами, хранящимися в репозитории, после чего все модифицированные файлы будут импортированы в репозиторий. Команда cvs update -d позволяет также импортировать все вновь созданные директории. Она сообщает о результатах своего выполнения в виде атрибутов статуса каждого файла:

- *U* Обновление файла завершилось успешно.
- *A* Файл добавлен, но изменения еще не зафиксированы (необходимо запустить команду cvs commit).
- *R* Файл удален, но изменения еще не зафиксированы (необходимо запустить команду cvs commit).
- *M* Файл модифицирован в вашей рабочей директории; файл в репозитории был изменен, в результате чего файл в рабочей папке оказался более старым, чем в репозитории (его обновление было выполнено до момента последней проверки системой CVS), либо в репозитории имеются изменения, безопасное слияние которых система произвести не может.
- *C* Конфликт между копиями файла в рабочей папке и репозитории, для разрешения которого требуется вмешательство оператора.

? Файл существует только в рабочей директории и отсутствует в репозитории; система CVS не знает, что ей с ним делать.

СОВЕТ №29

CVS: использование тегов
(Теги версии модуля в CVS)

Теги используются для присвоения символьного имени версии одного или нескольких файлов. Команда cvs tag позволяет задавать в репозитории теги для всех файлов и вложенных папок текущей рабочей директории. Применение команды cvs tag основано на использовании метода даты/времени. Она задает символическое имя для версии файла либо папки, ближайшей к заданной метке даты/времени; сама рабочая директория при этом не учитывается. Обратите внимание: система CVS не позволяет использовать в тегах символ «.». После названия тега вы обязаны задать имя файла, как в приведенном ниже примере:

```
cvs tag имя_тега имя_файла
cvs tag
```

В противном случае, данная команда присвоит теги всем файлам из текущей рабочей директории, хранящимся в репозитории:

```
cvs tag -c имя_тега
```

Применение опции -c позволяет прервать процесс, если копии файлов в репозитории отличаются от файлов в рабочей директории:

```
cvs/example$ cvs tag release-1-0 src/sample.c
cvs/example/src$ cvs tag release-1-0
cvs/example/src$ cvs tag -c release-1-0
```

Чтобы извлечь версии файлов с тегами, воспользуйтесь флажком -r во время выполнения проверки либо обновления. Если вы выполните проверку и обновление файлов внутри своей обычной рабочей директории, версии файлов с тегами заменят существующие версии:

```
cvs checkout -r имя_тега
cvs update -r имя_тега
```

Приведем пример:

```
cvs$ mkdir example-rel-1.0
cvs/example-rel-1.0$ cvs checkout -r release-1-0
```

либо

```
cvs/example$ cvs update -r release-1-0
```

СОВЕТ №30

CVS: Внесение изменений в модуль
(Проверка изменений в CVS)

После проверки файлов их требуется откомпилировать в обычном режиме. Зафиксируйте изменения в репозитории при помощи команды cvs commit. Она должна быть запущена, когда вы находитесь выше по иерархии каталогов, чем все модифицированные файлы — ее всегда можете запустить из рабочей папки.

Воспользуйтесь командой cvs commit имя_файла для фиксации изменений в одном файле либо рекурсивной фиксации папки со всеми подкаталогами. Различные группы разработчиков имеют собственное мнение по поводу того, как часто нужно выполнять команду cvs commit. Хорошее практическое правило гласит: «каждый раз, когда выполняется перекомпиляция проекта» и «каждый день до и после работы».

```
cvs/examples$ cvs commit
cvs commit: Examining .
cvs commit: Examining src
jenn@cvs.sample.com.au's password:
```

Система CVS анализирует каждую директорию со всеми вложенными поддиректориями внутри рабочего каталога. Все файлы, о существовании которых известно системе CVS, проверяются на наличие изменений. Если репозиторий находится на удаленном компьютере, CVS запросит пароль для доступа к нему, если вы предварительно не позаботитесь об использовании ключей ssh вместо паролей (см. раздел *«Быстрый вход в систему при помощи клиентских ключей ssh»* **[Совет № 66]**).

Затем система CVS откроет редактор, используемый по умолчанию в вашем окружении – в соответствии со значением переменных $CVSEDITOR и $EDITOR. Добавьте комментарии для соответствующих файлов:

```
CVS:----------------------------------------------------------------------
CVS: Журнал ввод. Строки, начинающиеся с 'CVS:', удаляются автоматически
CVS:
CVS: Фиксация в .
CVS:
CVS: Модифицированные файлы:
CVS: examples/src/sample.h examples/src/sample.c
CVS:----------------------------------------------------------------------
```

Обязательно добавляйте информативные примечания — иначе, когда понадобится выполнить откат изменений и натолкнувшись на примечание типа «исправил несколько ошибок», не сможете определить, до какой версии нужно осуществлять откат без помощи команды cvs diff. В случае возникновения потенциального конфликта выполнение

команды cvs commit будет прервано. Можно исправить ситуацию, применив к репозиторию команду cvs update, в результате чего будет произведена попытка слияния данных и запрошено вмешательство пользователя, если осуществить слияние без потерь невозможно:

```
cvs server: Up-to-date check failed for 'cvs_intro.html'
cvs [server aborted]: correct above errors first!
cvs commit: saving log message in /tmp/cvst7onmJ
```

СОВЕТ №31

CVS: слияние файлов

(Разрешение конфликтов обновления в CVS)

Когда система CVS не в состоянии успешно завершить слияние модифицированного файла с его копией в репозитории, она сообщает о конфликте в выходных данных системы cvs update. При этом оригинальный файл сохраняется в рабочем каталоге под названием *.#file.version*, а результаты слияния помещаются в файл с первоначальным именем:

```
cvs/examples$ cvs update
jenn@cvs.sample.com.au's password:
cvs server: Updating .
RCS file: /home/cvs/examle/sample.c,v
retrieving revision 1.3
retrieving revision 1.4
Merging differences between 1.3 and 1.4 into sample.c
rcsmerge: warning: conflict during merge
cvs server: conflicts found in sample.c
C sample.c
```

Система CVS сохраняет результаты слияния в строках, выделенных тегами CVS. CVS не может разрешить конфликт, когда одна и та же строка подверглась изменениям в обеих версиях файла:

```
<<<<<<<<< sample.c
Deliberately creating a conflict.
=======
Let's make a conflict.
>>>>>>>>> 1.4
```

СОВЕТ №32

CVS: добавление и удаление файлов и директорий

(Добавление и удаление файлов из модуля)

Файлы, с которыми система CVS не знает, что нужно делать, отмечаются знаком вопроса после процесса фиксации изменений и во время

выполнения команды cvs update. Чтобы система CVS могла распознавать в них изменения, вначале их необходимо добавить в репозиторий.

Для добавления в репозиторий новых файлов воспользуйтесь командой cvs add имя_файла. Ни один файл не будет реально добавлен в репозиторий системой CVS, пока вы не примените команду cvs commit.

Добавление директорий производится аналогичным способом. Причем файлы из добавляемой директории не могут быть помещены в репозиторий до того, как в хранилище будет добавлена сама директория.

Удаление файлов

Чтобы отметить файл для удаления из рабочих копий, воспользуйтесь командой cvs remove имя_файла. Перед удалением заданного файла из репозитория, его понадобится удалить из файловой системы. Система CVS в действительности не удаляет файл безвозвратно, просто перемещая его в специальный подкаталог под названием *Attic*.

Удаление каталогов

Директории не могут быть удалены из репозитория при помощи команд системы CVS. Если определенная директория больше не нужна, очистите ее содержимое при помощи команды cvs remove, а затем воспользуйтесь командой cvs update -P или cvs checkout -P во время обращения к рабочей копии. Наличие флажка -P гарантирует, что пустые директории не будут востребованы.

Если возникнет необходимость, можете удалить директорию при помощи команды rmdir, применяя ее непосредственно к репозиторию. Попробуйте вначале воспользоваться данной командой для копии репозитория и проверьте, не нарушено ли что-нибудь: если внутри данной директории, размещенной в хранилище, присутствует поддиректория *Attic*, можно потерять архивные копии удаленных ранее файлов.

При удалении директории из репозитория необходимо позаботиться, чтобы все пользователи удалили свои рабочие копии данной папки и проверили наличие обновленных копий модуля.

СОВЕТ №33

CVS: разработка с ветвлением

(Настройка разработки с ветвлением в CVS)

В том случае, когда требуется исправить ошибку в предыдущих версиях программного кода, не меняя текущий код, либо модифицировать настройки отладочных серверов, не затрагивая настройки рабочих серверов, необходимо разделить модуль на отдельные ветви. Такое раз-

деление позволит сохранять и извлекать версии основного модуля, не влияя на сам модуль. Изменения, внесенные в отдельную ветвь, могут быть подвергнуты слиянию позднее. Для создания ветви воспользуйтесь командой cvs tag -b тег_ветви:

```
cvs/examples$ cvs tag -b release-1-0-patches
```

Используйте команды cvs update или checkout для обращения к данной ветви. Команда checkout создаст для новой ветви новую директорию, а команда update перезапишет содержимое рабочей директории файлами из новой ветви.

```
cvs checkout -r тег_ветви
cvs update -r тег_ветви
```

Например:

```
cvs/example-rel-1.0$ cvs checkout -r release-1-0-patches
cvs/examples$ cvs update -r release-1-0-patches
```

Впоследствии можно будет произвести слияние изменений, внесенных в различные ветви, при помощи системы разрешения конфликтов, вызываемой командами cvs update и cvs commit.

```
cvs checkout имя_модуля
cvs update -j тег_ветви
```

Например:

```
/tmp/examples$ cvs checkout example
/tmp/examples$ cvs update -j release-1-0-patches
```

либо в виде одной команды:

```
cvs checkout -j тег_ветви имя_модуля
```

Тогда наш пример, который будет выглядеть следующим образом:

```
/tmp/examples$ cvs checkout -j release-1-0-patches example
```

позволит разрешить все конфликты, о которых сообщала система во время использования команды cvs commit.

СОВЕТ №34 CVS: наблюдение за файлами и их блокировка

(Настройка наблюдения за файлами в CVS при помощи электронной почты)

В отличие от других систем контроля версий в системе CVS нет блокировки файлов – в ней возможно одновременное редактирование файлов. Тем не менее можно установить наблюдение за рядом файлов, в результа-

те чего система CVS будет рассылать уведомления по электронной почте в случае начала редактирования файла. Если определенные файлы находятся под наблюдением, разработчикам понадобится прибегнуть к помощи команд cvs edit и cvs unedit для высвобождения файлов для редактирования. Файлы, за которыми наблюдение не ведется, могут редактироваться без необходимости каким-либо образом уведомлять систему CVS.

Чтобы настроить наблюдение за определенными файлами, воспользуйтесь командами:

cvs watch on (список файлов)
cvs watch off (список файлов)

Чтобы сделать себя наблюдателем, понадобится выполнить следующие команды:

cvs watch add (список файлов)
cvs watch remove (список файлов)

либо

cvs watch add -a edit|unedit|commit|all (список файлов)
cvs watch remove -a edit|unedit|commit|all (список файлов)

Специальный файл уведомлений CVS задает действия, выполняемые в случае внесения изменений в файл, за которым ведется наблюдение. По умолчанию, пользователю системы CVS отсылается уведомление на адрес электронной почты, начинающийся с имени пользователя на этом сервере. Если электронный почтовый адрес пользователя отличается, потребуется настроить файл users в корневом каталоге репозитория CVSROOT. Записи файла должны быть представлены в формате имя_пользователя: адрес_электронной_почты по одному пользователю в строке, например:

```
jenn:jenn@cvs.example.com.au
```

СОВЕТ №35

CVS: безопасность системы

(Защита пользователей и программного кода, поддерживаемого системой CVS)

Удаленные репозитории

Когда репозиторий размещен на локальной системе, доступ к нему и способы обеспечения безопасности достаточно наглядны — можно задать в переменной среды $CVSROOT путь к корневому каталогу хранилища CVS либо вызвать проверку при помощи опции -d имя_папки.

Если же репозиторий находится на удаленном компьютере, необходимо сообщить системе CVS, на каком именно компьютере он разме-

щен и какой метод должен использоваться для доступа к этой системе. Существует несколько возможных методов, однако в качестве максимально простого и безопасного лучше использовать ssh. Синтаксис задания доступа к удаленному репозиторию в переменной $CVSROOT таков: :метод:[[пользователь]:[пароль]@]имя_хоста[:[порт]]:/ путь/к/репозиторию. Например:

```
:ext:jenn@cvs.example.com.au:/usr/local/cvsroot
```

Информация о синтаксисе, выдаваемая командой info cvs, может отличаться от приведенной в зависимости от версии системы CVS. Применяйте синтаксис, работающий на вашей системе. Воспользуйтесь для работы через SSH методом ext. В этом методе используется внешняя для CVS программа rsh либо rsh-совместимая программа для общения с сервером CVS. Чтобы заставить CVS использовать SSH вместо rsh, задайте в переменной среды $CVS_RSH значение SSH. Настройте SSH на сервере и клиентах; убедитесь, что ключи SSH сгенерированы для всех пользователей, последние располагают учетными записями и паролями для доступа к обоим компьютерам. Если имена пользователей на двух компьютерах одинаковы, задавать часть строки имя_пользователя@ в переменной CVSROOT необязательно. Если используется стандартный порт SSH, параметр номер_порта также можно опустить.

```
cvs -d :ext:cvs.example.com.au:/usr/local/cvsroot checkout sample<code>
```

Разрешения

Файлы в репозитории предназначены только для чтения, и поэтому менять разрешения файловой системы для них не стоит. Для управления доступом следует использовать разрешения директорий. Большинство администраторов создают группы, включающие людей, которые обладают доступом к модулю и имеют право на запись в данной директории.

Если вы используете удаленный репозиторий, задайте для корневой папки модуля бит setgid, чтобы гарантировать для всех вложенных папок установку корректных разрешений. Если репозиторий размещен на локальной системе, для контроля над разрешениями для файлов и папок в репозитории следует использовать переменную $CVSMASK.

Компьютер разработчика

Под *защитой проекта* подразумевается обеспечение безопасности репозитория и всех проверяемых копий, как правило, расположенных

на компьютерах разработчиков. Недостаточно только обеспечить защиту репозитория и убедиться, что передача данных между ним и компьютерами разработчиков осуществляется в зашифрованном виде, если кто-то может прийти в выходной день в офис разработчика и скопировать программный код на компакт-диск. Необходимо позаботиться о банальной физической безопасности и безопасности сети между компьютерами разработчиков, защите прототипов и демонстрационных копий, а также любых других мест, из которых может осуществляться проверка кода.

СОВЕТ №36

CVS: анонимные репозитории

(Создание собственного анонимного репозитория CVS, доступного только для чтения)

Создание анонимного репозитория

Метод доступа pserver позволяет пользователю удаленно подключаться к репозиториям, задавая свое имя и пароль, которые сверяются с файлом паролей на системе CVS либо системным файлом */etc/passwd*. К сожалению, эти учетные данные передаются в незашифрованном виде, так что в лучшем случае злоумышленник сможет перехватить их для доступа к репозиторию CVS, а в худшем — скомпрометировать систему в целом. По этой причине большинство людей предпочитают использовать для доступа к удаленным репозиториям ssh. Более подробные сведения см. в разделе *«CVS: безопасность системы»* **[Совет № 35]**.

Очевидно, что если требуется предоставить общий доступ к дереву исходных кодов только для чтения, использование с этой целью ssh является излишней и непрактичной мерой. Здесь лучше использовать pserver для предоставления простого анонимного доступа к репозиторию. Но перед тем как разрешить в системе CVS анонимный доступ, необходимо настроить компьютер, на котором размещен репозиторий, на использование традиционного метода pserver.

Установка pserver

Модуль pserver используется для организации удаленного анонимного доступа к репозиторию CVS, поэтому нужно создать пользователя, который не будет иметь прав на запись ни в одну из папок репозитория. Создайте пользователя с именем anonymous, либо воспользуйтесь другим часто используемым именем — cvsanon. Задайте для него в качестве оболочки /bin/true, а в качестве домашней директории что-то безобидное, вроде */var/empty*; создайте отдельную группу и заблокируйте

его пароль (самый быстрый способ сделать это — команда passwd -l). Этот пользователь никогда не будет входить в систему интерактивно: мы просто создаем учетную запись-«заполнитель» для системы CVS, чтобы установить для нее в дальнейшем бит setuid.

Создайте файл пароля для системы CVS. Добавьте следующую строку в файл с именем *CVSROOT/passwd* внутри вашей папки репозитория:

```
anonymous:23MLN3ne5kvBM
```

Если вы решили назвать анонимного пользователя cvsanon, видоизмените строку следующим образом:

```
anonymous:23MLN3ne5kvBM:cvsanon
```

В файле паролей системы CVS (*passwd*) в строке, слева направо, содержится следующая информация: имя учетной записи CVS, зашифрованный пароль и имя системной учетной записи, связанной с учетной записью CVS (последний параметр необязателен). Если эта строка будет опущена, CVS воспользуется первой строкой из системного файла паролей. В качестве зашифрованной строки представлен пароль anonymous. Чтобы быть абсолютно уверенным в том, что анонимный пользователь никогда не сможет внести изменения в репозиторий, добавьте строку anonymous в файл *CVSROOT/readers* внутри репозитория. Этот файл определяет всех занесенных в него пользователей как имеющих права только на чтение.

Теперь нужно позаботиться о том, чтобы система CVS не позволяла обычным системным пользователям подключаться с помощью метода pserver, для предотвращения привычного использования системных учетных записей через метод pserver своенравными пользователями. Эта настройка задается в файле *CVSROOT/config* внутри папки репозитория. Уберите символы комментариев в начале строки SystemAuth=no, после чего только пользователи, перечисленные в файле *CVSROOT/passwd*, смогут подключаться к хранилищу с помощью метода pserver. Обратите внимание: данный прием не возымеет эффекта для пользователей CVS, использующих для обращения к репозиторию методы ext и ssh; для контроля доступа через эти методы будут применяться простые файловые разрешения, и при этом никогда не будет производиться обращение к файлу *passwd* системы CVS.

И, наконец, нужно заставить систему на самом деле принимать подключения pserver. Дело в том, что система CVS не может выполняться в режиме демона: ее необходимо запускать из inetd. Она работает с портом 2401, поэтому понадобится добавить в файл */etc/services* следующую строку:

```
pserver 2401/tcp
```

И в файл */etc/inetd.conf*:

```
pserver stream tcp nowait root /usr/bin/cvs cvs --allow-root=/usr/local/
cvsroot pserver
```

Подставьте имя каталога репозитория вместо */usr/local/cvsroot*. Выполните команду skill -HUP inetd или обратитесь к разделу *«Начинаем работу с системой RCS»* **[Совет № 23]**, если команда skill не установлена, и продолжайте работу.

Применение удаленного доступа через pserver

Для тестирования своего анонимного репозитория задайте сначала следующее значение для переменной $CVSROOT:

```
:pserver:anonymous@your.machine.here:/usr/local/cvsroot
```

Перед выполнением команды cvs checkout необходимо подключиться с помощью метода pserver, запустив команду cvs login. На запрос системы о пароле введите **anonymous**. Теперь можно запускать команду cvs checkout имя_модуля и видеть обновления, как обычно. Убедитесь, что применить к репозиторию команду cvs checkin нельзя, и смело предоставляйте анонимный общий доступ к своему репозиторию.

Резервное копирование

Советы № 37-44

Рано или поздно наступает день, когда случается непредвиденное. Заканчивается срок службы жесткого диска. Пленка растягивается и рвется. Возможно, вы по ошибке задаете в команде не тот переключатель (либо делаете это, находясь в другой директории, на другом компьютере) и в следующий момент проклинаете себя за нажатие клавиши Enter, наблюдая, как с диска исчезают все ваши данные. Только в такие мгновения становится очевидной истинная ценность надлежащим образом спроектированных и реализованных решений по резервному копированию.

Хотя в этом разделе представлено несколько интересных подходов к резервному копированию данных, его ни в коем случае нельзя считать содержащим исчерпывающую информацию. Выступая в роли системного администратора, вы должны написать план резервного копирования, удовлетворяющий нуждам вашей организации, реализовать его и при необходимости регулярно пересматривать. Не следует полагаться на то, что, задав в планировщике выполнение резервного копирования по расписанию, вы гарантированно получите работоспособные резервные копии, даже если сам процесс резервного копирования пройдет без ошибок. Необходимо следить за журналами событий, приобретать новые носители информации до того, как старые начнут вызывать проблемы. Напишите политику, в соответствии с которой будет выполняться тестирование резервных копий путем полного восстановления данных из них настолько часто, насколько считаете это необходимым, чтобы обеспечить себе спокойный сон. И даже в этом случае обдумайте возможность хранения дополнительной копии за пределами основного офиса. Если вас серьезно интересует подробное описание всего, что связано с построением безопасной политики резервного копирования данных, обратитесь к книге «Резервное копирование и восстановление Unix» В. Куртиса Престона (издательский дом O'Reilly).

В этой главе представлен ряд методов хранения данных в полной готовности к «судному дню». Каждый установленный экземпляр операционной системы Unix обладает своими уникальными характеристиками, и, кроме того, для каждого сервера необходим свой уровень резервирования данных, начиная от сохранения изменений, внесенных в течение недели, и заканчивая мгновенными копиями состояния, создаваемыми каждые 5 минут. Поэтому здесь рассматриваются некоторые инструментальные средства, с помощью которых вы сможете разработать свою собственную политику.

СОВЕТ №37

Резервное копирование с помощью tar в ssh

(Копирование отдельных файлов и папок между серверами с помощью ssh и tar)

Обмен файлами между серверами легко выполнить при помощи команды scp:

```
root@inky:~# scp some-archive.tgz blinky:/
```

либо даже копировать по нескольку файлов за один раз:

```
root@pinky:~/tmp# scp clyde:/usr/local/etc/*
```

Однако команда scp не подходит для копирования поддиректорий с сохранением файловых разрешений и прав собственности. К счастью, существует утилита tar, которая является одним из наиболее старых проектных решений в ssh, позволяющих этой утилите выполняться как любой другой стандартной команде Unix. Если использовать ее для выполнения команд вне интерактивной сессии работы в системе, ssh просто будет принимать данные в STDIN и выводить результаты в STDOUT. Любой канал, в котором задействована служба ssh, можно рассматривать как простой портал к компьютеру, с которым вы связаны. Предположим, требуется создать резервные копии всех домашних директорий на одном сервере и записать их в архив на другом сервере:

```
root@inky~# tar zcvf – /home |ssh pinky "cat > inky-homez.tgz"
```

или осуществить запись сжатого архива непосредственно на устройство резервного копирования на магнитной ленте, подключенное к удаленной системе:

```
root@blinky~# tar zcvf – /var/named/data | ssh clyde "cat > /dev/tape"
```

Предположим, нужно сделать только копию структуры каталогов одного сервера на файловой системе другого сервера. В нашем примере имеется работающая версия сервера Apache на одном сервере и не-

работоспособная копия на другом сервере. Попытаемся выполнить синхронизацию обеих копий:

```
root@clyde:~# cd /user/local
root@clyde:/usr/local# tar zcf – apache/ \
| ssh pacman "cd /usr/local; mv apache apache.bak; tar zpxvf -"
```

В результате выполнения операции папка */usr/local/apache/* при помощи упаковщика pacman будет помещена в файл */usr/local/apache.bak*. Затем точная копия папки */usr/local/apache/* пользователя clyde будет восстановлена с сохранением файловых разрешений и структуры каталогов. Поэкспериментируйте со сжатием (флажок z утилиты tar), но при этом учтите, что общее быстродействие зависит от скорости обработки информации на обеих системах, пропускной способности и загруженности сети и от применения сжатия в ssh.

Наконец, предположим, что на локальной машине имеется огромный архив и требуется восстановить из него информацию на удаленную систему без предварительного копирования (предположим, он имеет действительно огромный размер, и места достаточно только для извлеченных данных, но не для размещения на системе самого архива):

```
root@blinky~# ssh pinky "cd /usr/local/pacland; tar zpvxf -" \
< really-big-archive.tgz
```

либо, в качестве альтернативы, необходимо извлечь данные из другого места:

```
root@pinky:/usr/local/pacland# ssh blinky "cat really-big-archive.tgz" \
| tar zpvxf
```

Если во время создания либо извлечения данных из архива на удаленный компьютер возникнут проблемы, проверьте, не запущены ли на вашей системе какие-нибудь программы, осуществляющие вывод на терминал (смотрите файл *~/.bashrc* на удаленной системе). Если для вывода данных на терминал используется */usr/games/fortune* либо иная программа, лучше всего хранить ссылку на нее в *~/.bash_profile* или *~/.bash_login*, чем в *~/.bashrc*, т.к. интерес представляют сообщения программы fortune, только когда к ней подключены реальные пользователи, а совсем не в тот момент, когда по сетевому каналу передаются результаты удаленного выполнения команды. В этом случае можно попрежнему использовать переменные среды и любые другие команды из *~/.bashrc*, если эти команды гарантированно не станут отправлять какие-либо данные в каналы STDOUT или STDERR.

Использование ключей ssh избавляет от необходимости применения паролей, дополнительно облегчая копирование и перенос произвольных файлов, а также добавление заданий в планировщик Unix. В сле-

дующих разделах показано, как подключаться к любым серверам с помощью ssh.

Дополнительная информация:

- *«Быстрый вход в систему при помощи клиентских ключей ssh»* **[Совет № 66]**;
- *«Подключение через ssh в турборежиме»* **[Совет № 67]**;
- *«Эффективное использование агента ssh»* **[Совет № 68]**.

СОВЕТ №38 Использование rsync в ssh

(Быстрая синхронизация структуры больших каталогов с помощью rsync)

В то время как утилита tar идеально подходит для использования в ssh с целью создания удаленных копий отдельных фрагментов файловой системы, утилита rsync лучше всего подходит для синхронизации файловых систем между двумя компьютерами. Как правило, tar используется для создания первоначальной копии, а rsync — для копирования изменений, внесенных со времени создания последней копии. Дело в том, что tar выполняет копирование быстрее, чем утилита rsync, когда ни один из файлов в пункте назначения не существует, но rsync действует гораздо быстрее, если между двумя файловыми системами присутствуют лишь незначительные различия.

Чтобы запустить команду rsync через ssh, необходимо задать в командной строке переключатель -e:

```
root@rover:~# rsync -ave ssh greendome:/home/ftp/pub/ /home/ftp/pub/
```

Обратите внимание на завершающий символ / при задании исходного файла на сервере greendome. Такой способ указания источника для копирования сообщает программе rsync о необходимости копирования *содержимого* директории, но не самой директории. Если хотите скопировать саму директорию со всем ее содержимым, задайте исходный путь без символа / в конце:

```
root@village:~# rsync -ave ssh bcnu:/home/six .
```

Таким образом, копия директории *~root/six/* будет синхронизирована с оригиналом, размещенным на *bcnu:/home/six/*.

По умолчанию, утилита rsync только копирует файлы и директории, но не удаляет их с получателя в случае удаления файлов с источника. Чтобы гарантировать полную идентичность содержимого файлов папок на источнике и получателе, необходимо включать флажок –delete:

```
six@jammer:~/public_html# rsinc -ave ssh -delete greendome:~one/reports .
```

Теперь, при удалении старых отчетов из папки *~one/reports* на компьютере greendome, они будут автоматически удаляться из папки *~six/public_html/reports/* на компьютере jammer при каждом выполнении этой команды. Если запускаете команду из планировщика, уберите переключатель v, тогда она будет выполняться без лишнего шума, а в случае возникновения проблем с программой rsync отчет об ошибках придет по электронной почте.

Использование ssh в качестве коммуникационного протокола для трафика rsync имеет дополнительные преимущества, т.к. передаваемые по сети данные шифруются. Кроме того, появляется возможность использования всех предварительно установленных при помощи клиентских ключей ssh доверительных взаимоотношений. Для синхронизации объемных директорий со сложной структурой, особенно если между ними немного различий, утилита rsync является очень удобным и быстрым инструментальным средством, которое всегда должно быть в вашем распоряжении.

Дополнительная информация:

- оперативное руководство по утилите rsync (команда man rsync);
- *«Быстрый вход в систему при помощи клиентских ключей ssh»* **[Совет № 66]**;
- *«Эффективное использование агента ssh»* **[Совет № 68]**;
- *«Автоматизация процесса добавочного копирования с помощью программы rsync»* **[Совет № 42]**.

СОВЕТ **№39**

Архивация с помощью утилиты pax

(Создание простых машинонезависимых архивов с помощью утилиты pax)

Название утилиты pax расшифровывается как «portable archive exchange – обмен машинонезависимыми архивами». Она была специально разработана для обеспечения переносимости архивов между различными версиями Unix. Причем название было присвоено не без доли юмора, поскольку именно эта утилита стала символом примирения после долгой борьбы между сторонниками двух различных архиваторов: tar и cpio (в переводе с английского pax означает мир, символ мира). Данная утилита может применяться для создания любого из двух вышеперечисленных типов архивов, а при извлечении из архива автоматически определять его тип. Мы начнем изучение утилиты pax с базовых примеров, а затем перейдем к рассмотрению более интересных моментов, связанных с ее использованием.

Создание архивов

Для создания резервной копии своей домашней директории активизируйте режим записи при помощи переключателя w:

```
cd
pax -wf home.pax .
```

В этом примере переход в домашний каталог происходит при помощи команды cd, затем утилита pax получает команду на запись w в файл f под именем *home.pax* содержимого текущей директории (.). При использовании этой утилиты не забывайте включать переключатель f для определения имени файла архива, в который будет осуществляться запись. Если флажок f не будет установлен, различные печатаемые и непечатаемые символы начнут выводиться прямо на экран в сопровождении какофонических звуков. Убедитесь в том, что pax выполняет заданную работу, добавив переключатель v (сокр. от англ. verbose — многословный) к другим переключателям команды.

Чтобы увидеть, какой тип файла только что был создан, воспользуйтесь командой file:

```
file home.pax
home.pax: POSIX tar archive
```

Чтобы просмотреть содержимое архива, задайте в команде pax имя нужного архива при помощи переключателя f:

```
pax -f home.pax |more
```

Если архив имеет большой размер, задайте вывод результатов с помощью команды more, чтобы просматривать содержимое по одной странице за раз. Если зададите переключатель v, содержимое архива будет выводиться в таком же виде, как при выполнении команды ls -l. Не забудьте указать имя архива при помощи переключателя f, иначе ничего не произойдет, за исключением того, что вы не увидите приглашение командной строки, пока не нажмете Ctrl+C.

Распаковка архивов

Чтобы распаковать архив либо включить режим чтения, при помощи команды cd перейдите в папку, выступающую в роли получателя, а затем воспользуйтесь переключателем r команды pax. Например, извлечь содержимое резервной копии с именем *home.pax* в поддиректорию *test* внутри домашнего каталога можно следующим образом:

```
cd test
pax -rvf ~/home.pax
```

Утилита pax позволяет восстанавливать архивы tar и cpio. Она может автоматически определять корректный формат архива; тем не менее, рекомендуем предварительно всегда использовать утилиту file перед попыткой восстановления данных, чтобы проверить, является ли файл сжатым. Если это так, понадобится включить в команду переключать z.

В качестве примера возьмем файл с именем *backup.old*, размещенный в домашней директории (~). Вначале воспользуемся утилитой *file*:

```
file backup.old
backup: gzip compressed data, deflated, last modified:
Sat Aug 17 14:21:22 2002, os: Unix
```

Поскольку резервная копия является сжатой, то для ее распаковки в папку *test* нужно воспользоваться следующей командой:

```
cd test
pax -rvzf ~/backup.old
```

Кроме того, в домашнем каталоге имеется еще один файл с именем *backup*:

```
file: ~/backup
backup: cpio archive
```

Этот файл не является сжатым, поэтому его можно распаковать с помощью следующей команды:

```
pax -rvf ~/backup
```

Тот факт, что формат файла первой резервной копии оказался tar, а второй – cpio, не смутил утилиту pax; тем не менее, могло появиться сообщение об ошибке, если бы pax не была проинформирована о том, что файл первой резервной копии являлся сжатым.

Интерактивное восстановление

Выполняя восстановление из архивов при помощи утилиты pax, можно делать некоторые весьма интересные вещи, например, организовать интерактивное переименование/восстановление, включив в командную строку переключатель -i. Для этого понадобится задать следующую команду:

```
pax -rif ~/backup
```

В результате начнется интерактивная распаковка архива с именем backup в текущую директорию. В интерактивном режиме pax будет выводить имя каждого файла с запросом, не нужно ли переименовать

данный файл, восстановить его под тем же именем либо пропустить, не восстанавливая его:

```
ATTENTION: pax interactive file rename operation.
drwxr-xr-x Aug 17 15:08.
Input new name, or a "." to keep the old name, or a "return" to skip this file.
/* Введите новое имя, символ ".", чтобы оставить имя без изменений,
либо нажмите клавишу "return", чтобы пропустить файл.
Input >
Skipping file.
```

В данном случае нажмем клавишу Enter, поскольку не хотим менять имя «.» либо текущую директорию:

```
ATTENTION: pax interactive file rename operation.
drwxr-xr-x Aug 17 15:08.
Input new name, or a "." to keep the old name, or a "return" to skip this file.
Input > old
Processing continues, name changed to: old.

ATTENTION: pax interactive file rename operation.
drwxr-xr-x Aug 17 15:08.
Input new name, or a "." to keep the old name, or a "return" to skip this file.
Input >.
Processing continues, name unchanged.
```

Обратите внимание на то, что имя файла *file1* изменено на *old*, а *file2* — оставлено без изменений. В результате вывод листинга восстановленной директории покажет наличие двух файлов: *old* и *file2*.

Рекурсивное копирование директорий

Одна из наиболее мощных функций утилиты pax заключается в возможности быстрого копирования полной структуры каталогов с одного раздела жесткого диска на другой в режиме копирования. Чтобы воспользоваться режимом копирования, необходимо:

1. Перейти с помощью команды cd в папку-источник.
2. Убедиться в том, что папка-получатель существует; если нет — воспользоваться командой mkdir для ее создания.
3. Запустить на выполнение следующую команду: pax -rw . *папка_получатель*.

Обратите внимание: переключатель f в режиме копирования не задается, поскольку создавать архивный файл нам не нужно. Вместо этого в папке назначения напрямую воссоздается старая структура каталогов источника. К тому же запомнить синтаксис данной команды гораздо проще, чем ее эквивалента в tar:

```
tar cf – . | (cd папка_получатель; tar vpxf -)
```

А теперь посмотрите, что **делать не нужно**:

```
cd
mkdir test
pax -rw . test
```

В этом примере команда cd возвращает нас в домашний каталог, создает в нем поддиректорию под названием *test* и активизирует режим копирования. Таким образом, запускается бесконечный цикл создания вложенных директорий *test*, в каждую из которых будет помещено содержимое домашнего каталога. Если не прервать этот цикл нажатием клавиш Ctrl+C, pax будет продолжать работу до бесконечности, под которой в данном случае подразумевается исчерпание свободного дискового пространства. Именно на этот момент обратите внимание в оперативном руководстве по команде pax.

Внимание! Папка-получатель не должна входить в состав операндов команды либо являться вложенной папкой для какого-либо каталога, заданного в качестве операндов команды. При несоблюдении этих условий копирование чревато непредсказуемыми последствиями.

А вот следующая команда будет выполнена без проблем и практически мгновенно:

```
su
Password:
cd ~user1/big_project
mkdir ~user2/big_project
chown user2 ~user2/big_project
pax -rw . ~user2/big_project
```

Итак, структура каталога *big_project* полностью скопирована в домашний каталог другого пользователя. При использовании режима копирования необходимо обладать правами администратора, поскольку вам понадобится выполнять копирование за пределами своего домашнего каталога, а так удастся избежать ситуации с бесконечным циклом. Если понадобится создать новую папку, ее владельцем будет назначен пользователь root; при необходимости можете воспользоваться командой chown, чтобы гарантировать корректные полномочия на владение папкой перед началом копирования. Обратитесь к оперативному руководству по команде pax, чтобы разобраться, как управлять разрешениями для копируемой структуры каталогов.

Можно организовать интерактивное копирование структуры каталогов, включив в состав команды переключатель i:

```
pax -rwi . ~user2/big_project
```

Аналогично предыдущему примеру интерактивного использования, pax будет запрашивать действия для каждого файла, так что вы сможете определить, какие файлы необходимо будет копировать, а какие – переименовывать, если возникнет такая необходимость.

Добавочное резервное копирование

Применим команду pax с большей пользой, для чего продемонстрируем использование утилиты pax для создания системы добавочного резервного копирования. В данном примере, пользователь genesis собирается ежедневно выполнять резервное копирование всех изменений, произведенных в домашнем каталоге.

Во-первых, необходимо получить права администратора для создания директории, в которой будет сохраняться резервная копия:

```
su
Password:
mkdir /usr/backups
```

Затем создадим внутри нее вложенную директорию и присвоим права на владение ею пользователю genesis:

```
mkdir /usr/backups/genesis
chown genesis /usr/backups/genesis
```

Сменив свою учетную запись администратора на учетную запись пользователя genesis, выполним команду cd, чтобы вернуться в свою домашнюю директорию:

```
exit
cd
```

После чего выполняем полное резервное копирование своей домашней директории и сохраняем ее в архивный файл под именем *Monday*:

```
pax -wvf /usr/backups/genesis/Monday.
```

Теперь, имея в своем распоряжении полную резервную копию, можно выполнять добавочное копирование, отражающее только изменения, которые были внесены за день. Таким образом, закончив работу во вторник, запустим команду:

```
pax -wv -T 0000 -f /usr/backups/genesis/Tuesday.
```

Обратите внимание: в командной строке указан временной переключатель (-T) и время (0000 – полночь). Таким образом, утилита pax будет выполнять резервное копирование только тех файлов, в которые были внесены изменения после полуночи, т.е. всех файлов, модифи-

цированных за сегодняшний день. В среду нужно будет запустить точно такую же команду, только поменяв имя архива на *Wednesday*.

Если на диске достаточно пространства и вы собираетесь хранить резервные копии дольше недели, используйте в качестве архивных имен что-то вроде Aug01, Aug02 и т.п. Желательно в любом случае создавать полную резервную копию хотя бы раз в неделю, а затем уже ежедневно делать добавочные резервные копии. Если же объем дискового пространства ограничен, используйте переключатель z для сжатия файла резервной копии и экономии пространства на диске. Учтите также, что переключатель T может быть более придирчивым к параметрам, чем было продемонстрировано в предыдущем примере. Более подробную информацию можно получить в оперативном руководстве по утилите pax.

Пропуск файлов во время восстановления

Чтобы восстановить из резервной копии все файлы, кроме *file3*, воспользуйтесь следующей командой:

```
pax -rvf ~/backup -c './file3'
```

Переключатель *c* представляет собой переключатель исключения. Обратите внимание: шаблон исключений, в нашем примере file3, необходимо заключать в одинарные кавычки (клавиша рядом с Enter). Используйте либо символьный шаблон, известный для утилиты pax под именем ./file3, а не просто file3, либо специальный групповой символ, например такой:

```
pax -rvf ~/backup -c '*file3'
```

Если задать специальный символ (*) в начале шаблона, как показано в предыдущей строке, из процесса копирования будут исключены все файлы, заканчивающиеся на file3 (например, *file3, myfile3, thatfile3*).

И наоборот, можно перечислить только те файлы, которые должны быть извлечены при помощи переключателя n, используя шаблон соответствия. В результате выполнения следующей команды будет восстановлен только файл *file2*:

```
pax -rvf ~/backup -n' /.file2'
```

Переключатель *n* отличается от переключателя *c* тем, что при задании специальных групповых символов будет восстановлен только первый найденный файл, который соответствует шаблону. То есть, в результате выполнения следующей команды:

```
pax -rvf ~/backup -n '*file3'
```

из списка файлов *file3, myfile3* и *thatfile3* будет восстановлен только первый файл, соответствующий заданному шаблону, — файл *file3*.

Переключатели *c* и *n* могут пригодиться и во время создания архивов; используйте их, чтобы определить, какие файлы надо помещать в архив, а какие — нет.

Дополнительная информация:

- оригинальная статья, размещенная по адресу: *http://www.onlamp.com/pub/a/bsd/2002/08/22/freeBSD_Basics.html*;
- исходные коды утилиты pax, представляющие собой часть открытого пакета на веб-странице *http://www.research.att.com/sw/download/*.

СОВЕТ №40

Резервное копирование загрузочного раздела

(Храните резервную копию своего загрузочного раздела в надежном месте на случай непредвиденных обстоятельств)

Установка загрузчика, например, LILO, может оказаться сложнее, чем того бы хотелось. Особенно в случае использования IDE-устройств, можно попасть в ситуацию, когда загрузиться без использования резервного диска окажется просто невозможно.

Одна из наиболее распространенных ошибок, связанных с использование жестких дисков IDE, заключается в установке ядра на раздел, охватывающий более 1024 цилиндров. Симптомы в этом случае могут оказаться весьма странными, поскольку вначале компьютер будет загружаться без проблем, а позднее, после установки ядра и того, как компьютер поработает некоторое время, LILO начнет выдавать ошибку, а впоследствии откажется загружаться. Подобные симптомы могут поставить администратора в тупик, поскольку «все ведь работало». Скорее всего, при первой установке ядра оно оказалось размещенным на поверхности диска, лежащей до 1024-го цилиндра. Однако после доустановки системного и другого программного обеспечения, диск начал заполняться. Когда диск будет содержать около 500 Мб (либо 1 и более Гб на некоторых версиях BIOS) данных, все новые версии ядра окажутся (по крайней мере, частично) за пределами ограничения в 1024 цилиндра — что сделает его недоступным для BIOS во время загрузки.

Последние версии LILO откажутся устанавливать ядро при таких условиях, тогда как в ряде предыдущих версий пользователю просто выдавалось предупреждение, после чего установка продолжалась обычным образом. Один из способов решения этой проблемы заключается в создании небольшого, скажем, 10-мегабайтного, раздела перед началом установки — в качестве */dev/hda1*, первого раздела на диске — и его установка под именем */boot*. Затем при установке новых версий ядра придется всегда копировать их в раздел */boot*, и тогда они гарантированно будут работать с вашей версией BIOS, поскольку не нарушат ограничения в 1024 цилиндра.

В любом случае к установке загрузчика не следует относиться легкомысленно. Как только закончите установку ядра и заставите его работать так, как того требовалось, постарайтесь создать резервную копию всего загрузочного раздела своего загрузочного диска.

Для IDE-устройств необходимо выполнить следующую команду:

```
dd if=/dev/hda of=bootsector.bin bs=512 count=1
```

Для SCSI-устройств понадобится немного другая команда:

```
dd if=/dev/sda of=bootsector.bin bs=512 count=1
```

И, в качестве альтернативы, можете скопировать загрузочный сектор непосредственно на гибкий магнитный диск:

```
dd if=/dev/hda of=/dev/fd0 bs=512 count=1
```

Очень внимательно задавайте операнды if и of, поскольку, перепутав их, вы уничтожите загрузочный сектор системы. Лучше сохранить загрузочный сектор в файл, а затем скопировать его на сменный носитель, как в случае с любым другим файлом. Запись критичных системных данных на дискету может показаться хитрым решением, но после того как первая же (или тысячная) дискета подведет в самый ответственный момент, вы наверняка откажетесь от своего пристрастия к гибким дискам.

Теперь, если случайно запортится загрузочный сектор, его можно быстро восстановить при помощи команды:

```
dd if=bootsector.bin of=/dev/hda
```

либо, если загрузочный сектор все-таки скопирован на дискету:

```
dd if=/dev/fd0 of=/dev/hda bs=512 count=1
```

Обычно, если вместо IDE-устройств используются устройства SCSI, в вышеперечисленных командах следует заменять sda на hda.

СОВЕТ №41

Синхронизация отдельных частей файловой системы при помощи утилиты rsync

(Используйте утилиту rsync через ssh для дублирования нужной части файловой системы на любом количестве серверов)

Для системы веб-публикаций издательства O'Reilly нами был создан кластер веб-серверов, позволяющий распределять работу между несколькими компьютерами, повышая общую производительность и надежность. Однако как синхронизировать реальные данные файловой системы между несколькими серверами?

Один из методов сводится к использованию NFS (Network File System – сетевой файловой системы) для хранения общих данных. Хотя NFS уп-

рощает одновременное обновление информации на всех серверах, она обладает рядом существенных недостатков. Так, например, печально известны сложности с блокировкой файлов в NFS. Настройка производительности NFS вообще считается чем-то сродни черной магии. (Если она работает, это очень хорошо, если же нет — мы к этому готовы). Но, вероятно, самым существенным недостатком является наличие монолитного сервера NFS, сбой которого может заблокировать работу всей организации. Если сервер NFS окажется недоступен по какой-либо причине, от этого пострадают все зависимые машины.

В качестве альтернативного подхода применяется асинхронное обновление серверов при помощи утилиты rsync. С падением стоимости единицы дискового пространства имеет смысл выполнять локальное кэширование данных: причем не только с точки зрения отказоустойчивости, но и с точки зрения быстродействия — все-таки доступ к локальным файлам выполняется намного быстрее, чем к файлам, размещенным где-то на сетевом хранилище.

Добавьте в планировщик задание для запуска утилиты rsync на каждом сервере:

```
rsync -ae ssh master.machine.com:/usr/local/apache/htdocs/ \
/usr/local/apache/htdocs/
```

Полагая, что вы уже успели предварительно настроить свои ключи ssh (см. раздел *«Быстрый вход в систему при помощи клиентских ключей ssh»* **[Совет № 66]**), с помощью этой команды сможете синхронизировать содержимое локальной корневой папки с документами Apache с текущей копией, размещенной на *master.machine.com*, через зашифрованную сессию ssh. Поскольку изменения всегда сохраняются на *master.machine.com*, то при следующем проходе информация будет скопирована на все остальные серверы.

Наибольшим недостатком данного подхода является асинхронность обновлений. Если настроить выполнение задания на каждые 5 минут, копия может быть устаревшей, в среднем, минуты на три. Если приложение, с которым вы работаете, может смириться с такой ситуацией (буду через минуту), характерной для утилиты rsync при копировании больших файловых структур, этот метод поможет выиграть в терминах реальной производительности и надежности сервера. Единственное строгое предупреждение: необходимо позаботиться о том, чтобы до начала выполнения следующего задания с участием утилиты rsync было закончено выполнение предыдущего. В противном случае, если сервер слишком загружен, можно оказаться в ситуации так называемой «нисходящей спирали», когда запуск нового задания с утилитой rsync будет увеличивать загрузку сервера, что, в свою очередь, вызовет замедление работы rsync, в результате чего, когда придет очередь выполнять следу-

ющую синхронизацию, предыдущая может еще не закончиться, что еще более увеличит загрузку сервера, и т.д.

Тем не менее rsync в данной ситуации обладает рядом преимуществ перед tar. Запуск утилиты tar через ssh (см. раздел *«Резервное копирование с помощью tar в ssh»* **[Совет № 37]**) происходит быстрее в случае необходимости выполнять копирование полного дерева каталогов «с нуля», но утилита rsync действует намного быстрее, когда в основной копии было изменено всего несколько файлов. При первом проходе выполняется анализ файлов, подвергшихся изменениям, и только затем начинается копирование модифицированных файлов.

Но, предположим, к нашему веб-кластеру предъявлены более строгие требования. Допустим, мы пытаемся распределить нагрузку (см. раздел *«Распределение нагрузки при помощи директивы RewriteMap сервера Apache»* **[Совет № 99]**) по обслуживанию страниц между двумя группами: группой серверов приложений, работающих под управлением Apache с включенной опцией mod_perl, управляющей нашей системой распределения содержимого, и группой облегченных клиентских серверов Apache, занятых обслуживанием статичных данных, например файлов изображений и архивов. В данной ситуации копирование полного дерева документов на все серверы будет напрасной тратой дискового пространства, поскольку клиентские компьютеры должны обслуживать запросы только к определенным типам файлов.

Воспользовавшись функцией exclude-from (исключить из) утилиты rsync, задайте файл, определяющий, какая часть файловой системы будет обрабатываться rsync во время синхронизации. Создайте файл, подобный рассмотренному ниже */usr/local/etc/balance.front*, для своего статического сервера:

```
### То, что мы не хотим синхронизировать
#
- logs/
### весь корневой каталог с документами
#
+ /usr/
+ /usr/local/
+ /usr/local/apache/
+ /usr/local/apache/htdocs/
+ /usr/local/apache/htdocs/**

### пользовательские директории public_html
#
+ /home/
+ /home/*/
+ /home/*/public_html/
+ /home/*/public_html/**

# Рисунки, архивы и т.п.
```

```
#
+ *.jpg
+ *.gif
+ *.png
+ *.pdf
+ *.mp3
+ *.zip
+ *.tgz
+ *.gz

### Исключаем все остальное.
#
- *
```

Создадим подобный файл для синхронизации с группой серверов приложений:

```
### То, что мы не хотим синхронизировать
#
- logs/
- *.tmp
- *.swp

### весь корневой каталог с документами
#
+ /usr/
+ /usr/local/
+ /usr/local/apache/
+ /usr/local/apache/htdocs/
+ /usr/local/apache/htdocs/**

### Исключаем все остальное.
#
- *
```

Теперь напишите список серверов по типам — внешних и внутренних – по одному компьютеру в строке. Присвойте файлам имена */usr/local/etc/servers.front* и */usr/local/etc/servers.back*, соответственно.

К примеру, занесите в список *servers.front* такие серверы:

```
tiberius
caligula
```

а в *servers.back* — следующие:

```
augustus
claudius
germanicus
posthumous
castor
```

И, наконец, вместо того чтобы вызывать утилиту rsync непосредственно из планировщика на каждом своем веб-сервере, попробуйте за-

пустить из планировщика Unix приведенный ниже программный код на языке Perl.

Листинг: Balance-push.sh

```
#!/bin/bash

#
# balance-push – выполняет параллельный перенос содержимого с ведущего
# сервера (localhost) на клиентские и серверные системы.
#

# $FRONT_END – содержит список систем, выступающих в роли клиентов
# и получающих только обновления статического содержимого.
#
FRONT_END=$(cat /usr/local/etc/servers.front)

# $BACK_END – содержит список систем, выступающих в роли серверов
# и получающих полные обновления.
#
BACK_END=$(cat /usr/local/etc/servers.back)

# $TARGET задает корневой каталог удаленной файловой системы,
# в который должна копироваться информация.
# Как правило, в реальных условиях в качестве корневого каталога
# задается /, но он может быть изменен, например
# на время проведения тестирования.
#
TARGET=/

# $EXCLUDE задает префикс имен файлов, исключаемых из обработки утилитой
# rsync. К примеру, если хотите исключить файлы
# /usr/local/etc/balance.front и /usr/local/etc/balance.back, задайте
# в этой переменной значение "/usr/local/etc/balance".
#
EXCLUDE=/usr/local/etc/balance

# $LOCK_DIR задает путь, по которому будут размещены
# заблокированные файлы.
#
LOCK_DIR=/var/tmp

######## За кулисами мы игнорируем функции оболочки. ########

PATH=/bin:/usr/bin:/usr/local/bin

lock () {
local lockfile="$LOCK_DIR/balance.$1.lock"
if [ -f $lockfile ]; then
if kill -0 $(cat $lockfile); then
echo "$0 appears to be already running on $1."
# $0 уже запущен на $1
echo "Please check $lockfile if you think this is in error."
```

```
# Пожалуйста, проверьте $lockfile, если вы считаете,
#что это ошибка
exit 1
else
echo "$0 appears to have completed for $1 without cleaning up its lockfile."
# Процесс $0 завершил работу на $1, не сняв свои блокировки
fi
fi
echo $$ > $lockfile
}

unlock () {
rm -f $LOCK_DIR/balance.$1.lock
}

push_files () {
local mode=$1 host=$2

if [ ! "$mode" -o ! -r "$EXCLUDE.$mode" ]; then
echo "$0 $$: mode unset for $host!"
# режим не задан для узла сети
return
fi

if [ ! "$host" ]; then
echo "$0 $$: host unset for push $mode!"
# для использования данного режима не задан узел сети
return
fi

lock $host

rsync --archive --rsh=ssh --delete --ignore-errors --whole-file \
--exclude-from="$EXCLUDE.$mode" / ${host}:${TARGET}

unlock $host
}

push_tier () {
local mode=$1 host_list=$2

for host in $host_list; do
$SHELL -c "push_files $mode $host" &
done
}

export -f lock unlock push_files
export TARGET EXCLUDE LOCK_DIR PATH

[ "$FRONT_END" ] && push_tier front "$FRONT_END"
[ "$BACK_END" ] && push_tier back "$BACK_END"

#
# Конец сценария.
#
```

Этот сценарий, названный нами balance-push, берет на себя управление синхронизацией с помощью утилиты rsync, позаботившись о том, чтобы серверы не перекрывали друг друга во время синхронизации. Благодаря ему каждая группа серверов получит только те файлы, которые нужны данной группе, в соответствии с настройками, заданными в файлах */usr/local/etc/*. Если обнаружится, что на сервере еще не закончилось выполнение предыдущей синхронизации, повторный запуск синхронизации для него не будет произведен до тех пор, пока не завершится выполнение предыдущей (об этом будет сообщено по электронной почте с помощью функции планировщика MAILTO).

Разумеется, нагрузка на ваш основной сервер, скорее всего, вырастет, в зависимости от того, на скольких компьютерах требуется выполнить синхронизацию, поскольку для каждого сервера потребуется создание сессий rsync и ssh. Но на практике для рабочей сети, обслуживающей миллион обращений в день, нагрузку, создаваемую каждым веб-сервером по отдельности, можно даже не принимать в расчет.

СОВЕТ №42

Автоматизация процесса добавочного копирования с помощью программы rsync

(Применяйте утилиту rsync для быстрого создания защищенных мгновенных копий состояния своей файловой системы)

На серверах Linux существует метод автоматического создания резервных копий в стиле мгновенных копий состояния файловой системы. Функция резервного копирования мгновенного состояния файловой системы присуща некоторым файловым серверам промышленного масштаба: в них создается иллюзия ежедневного создания множества полных (вплоть до полномочий) резервных копий без перерасхода дискового пространства. Все подобные копии состояний доступны только для чтения, причем пользователи могут к ним обращаться напрямую через специальные системные директории.

Поскольку создание полной копии всей файловой системы – достаточно дорогостоящий и требовательный ко времени процесс, общепринятой практикой является создание полных резервных копий с периодичностью около раза в неделю или даже в месяц с сохранением в промежуточных резервных копиях только текущих изменений. Такая технология носит название «добавочного» резервного копирования, поддерживаемого утилитами типа dump и tar, и не только ими.

У команды cp, относящейся к стандартным файловым утилитам GNU, есть флажок -l, позволяющий создавать ссылки вместо копий (тем не менее она не создает привязки к директориям, что очень даже хорошо: подумайте, почему). Еще один удобный переключатель команды cp – -a (архив), при использовании которого осуществляется ре-

курсивная обработка всех вложенных директорий с сохранением информации о владельцах файлов, метках даты/времени и разрешениях файловой системы.

Комбинация флажков cp -al позволяет создавать нечто, имеющее вид полной копии дерева каталогов, а на самом деле представляющее собой иллюзию, практически не занимающую дискового пространства. Если ограничить операции, выполняемые над копией только добавлением и удалением файлов (ссылок), т.е. не модифицировать сами файлы, иллюзию полной резервной копии данных можно считать законченной. Для конечного пользователя единственное отличие будет состоять в том, что мнимая копия практически не будет занимать дискового пространства и будет создаваться почти мгновенно.

Мы можем использовать команды rsync и cp -al одновременно для создания нескольких мнимых полных резервных копий файловой системы, не тратя много дискового пространства:

```
rm -rf backup.3
mv backup.2 backup.3
mv backup.1 backup.2
cp -al backup.0 backup.1
rsync -a --delete source_directory/ backup.0/
```

Если запускать эту команду каждый день, тогда backup.0, backup.1, backup.2 и backup.3 будут выглядеть как полные резервные копии папки */исходная_папка/* по состоянию на сегодня, вчера, позавчера, позапозавчера, соответственно — за исключением того, что файловые разрешения и права владельцев в более старых копиях будут иметь последние действующие значения (спасибо Джею В. Шульцу за то, что обратил внимание на этот момент). На самом деле реальный размер дополнительного хранилища будет равен текущему размеру папки */исходная_папка/* плюс суммарный объем внесенных изменений за последние трое суток — что равноценно тому объему, который бы заняла полная резервная копия и добавочные копии, созданные при помощи утилит dump или tar.

Последний метод намного лучше подходит для создания резервных копий по сети, поскольку он требует однократного создания полной резервной копии вместо еженедельного копирования. После этого копироваться будут только изменения. К сожалению, утилиту rsync нельзя применять для создания резервных копий на магнитной ленте: для этого нужно обратиться за помощью к утилитам dump или tar.

Если у вас имеется незанятый компьютер, пусть даже не самый современный, его вполне можно превратить в выделенный сервер резервного копирования. Переведите систему в автономный режим работы и перенесите компьютер в место, физически удаленное от

основного офиса — хотя бы в другое помещение, а лучше — в другое здание. Отключите на сервере резервного копирования все удаленные службы и подключайте его к сети только через выделенный сетевой интерфейс исходного узла сети.

На этом сервере можно выполнять резервное копирование с помощью утилиты rsync через ssh (см. раздел *«Синхронизация отдельных частей файловой системы при помощи утилиты rsync»* **[Совет № 41]**) и экспортировать резервные копии обратно на исходный узел сети при помощи доступной только для чтения файловой системы NFS. В этом случае пользователи смогут получать доступ к мгновенным копиям состояния файловой системы самостоятельно — без необходимости вмешательства со стороны системного администратора, — но при этом не будут иметь прав на внесение изменений в резервные копии.

Это неплохой способ защиты, но можно создать два сервера резервного копирования для уверенности, что хотя бы один из них будет функционировать в автономном режиме.

Расширения: ежечасные, ежедневные и еженедельные мгновенные копии

Немного поэкспериментировав, вы научитесь создавать чередующиеся многоуровневые мгновенные копии. Например, мгновенные копии файловой системы, создаваемые каждые четыре часа, или три последние «суточные» копии, создаваемые каждые сутки около полуночи. Можно пожелать хранить даже еженедельные или ежемесячные мгновенные копии, в зависимости от нужд и доступного объема дискового пространства.

Ниже приведены два сценария: один создает новую мгновенную копию состояния файловой системы каждые 4 часа, а затем удаляет самую старую из четырех существующих копий, выполняя циклический сдвиг; второй запускается раз в сутки и сохраняет копию состояния файловой системы за сутки. Использовать утилиту rsync для создания мгновенных копий более высокого уровня не имеет смысла: вместо этого достаточно применить команду cp -al к текущей «почасовой» копии.

Чтобы настроить автоматическое создание мгновенных копий, необходимо добавить в таблицу планировщика (crontab) для пользователя root следующие строки:

```
0 */4 * * * /usr/local/bin/make_snapshot.sh
0 13 * * * /usr/local/bin/daily_snapshot_rotate.sh
```

Таким образом, сценарий make_snapshot.sh будет запускаться каждые 4 часа для создания «почасовых» копий, а сценарий daily_snapshot_rotate.sh будет выполняться ежесуточно в 13:00. Ниже приводится листинг этих сценариев.

Листинг: make_snapshot.sh

```
#!/bin/bash
# ————————————————————————————————————————
# Утилита для создания мгновенных копий состояния
# файловой системы с циклическим сдвигом
# ————————————————————————————————————————
# RCS info: $Id: make_snapshot.sh,v 1.6 2002/04/06 04:20:00 mrubel Exp $
# ————————————————————————————————————————
# Этот сценарий предназначен для создания резервных копий
# мгновенного состояния каталога /home при вызове сценария
# ————————————————————————————————————————

# ———————— системные команды, используемые в сценарии ————————————
ID=/usr/bin/id;
ECHO=/bin/echo;

MOUNT=/bin/mount;
RM=/bin/rm;
MV=/bin/mv;
CP=/bin/cp;
TOUCH=/bin/touch;

RSYNC=/usr/bin/rsync;

# ———————— локализация файлов ————————————————————————

MOUNT_DEVICE=/dev/hdb1;
SNAPSHOT_Rw=/root/snapshot;
EXCLUDES=/usr/local/etc/backup_exclude;

# ———————— сам сценарий ————————————————————————————

# убедитесь, что вы обладаете правами пользователя root
if (( `$ID -u` != 0 )); then { $ECHO "Sorry, must be root. Exiting..."; exit; } fi

# попытка монтирования файловой системы с правами Rw;
# в случае неудачи – выход
$MOUNT -o remount,rw $MOUNT_DEVICE $SNAPSHOT_Rw;
if (( $? )); then
{
$ECHO "snapshot: could not remount $SNAPSHOT_Rw readwrite";
# невозможно смонтировать $SNAPSHOT_Rw с правами на чтение и запись
exit;
}
fi;

# создание мгновенных копий каталога/home, удаляемых путем циклического
# сдвига (начиная с самых старых) – данный фрагмент должен носить более
# общий характер

# этап 1: удаляем самую старую копию, если таковая существует:
if [ -d $SNAPSHOT_Rw/home/hourly.3 ] ; then \
$RM -rf $SNAPSHOT_Rw/home/hourly.3 ; \
```

```
fi ;
# этап 2: выполняем сдвиг более новых копий на место более старых,
# по одной, если они существуют
if [ -d $SNAPSHOT_Rw/home/hourly.2 ] ; then \
$MV $SNAPSHOT_Rw/home/hourly.2 $SNAPSHOT_Rw/home/hourly.3 ; \
fi;
if [ -d $SNAPSHOT_Rw/home/hourly.1 ] ; then \
$MV $SNAPSHOT_Rw/home/hourly.1 $SNAPSHOT_Rw/home/hourly.2 ; \
fi;

# этап 3: создаем только жесткие ссылки (за исключением каталогов)
# в последней копии состояния файловой системы, если таковая существует
if [ -d $SNAPSHOT_Rw/home/hourly.0 ] ; then \
$CP -al $SNAPSHOT_Rw/home/hourly.0 $SNAPSHOT_Rw/home/hourly.1 ; \
fi;

# этап 4: запускаем rsync для создания актуальной копии файловой
# системы (обратите внимание: rsync ведет себя по умолчанию подобно
# cp --remove-destination, поэтому вначале ссылки на получатель будут
# удалены). В противном случае мы могли бы перезаписать другие
# резервные копии!
$RSYNC \
-va --delete --delete-excluded \
--exclude-from="$EXCLUDES" \
/home/ $SNAPSHOT_Rw/home/hourly.0 ;

# этап 5: обновим параметр mtime копии hourly.0, записав в него время
# создания текущей копии
$TOUCH $SNAPSHOT_Rw/home/hourly.0 ;

# и все это для каталога home.

# сделаем мгновенную копию доступной только для чтения

$MOUNT -o remount,ro $MOUNT_DEVICE $SNAPSHOT_Rw;
if (( $? )); then
{
$ECHO "snapshot: could not remount $SNAPSHOT_Rw readonly";
# Невозможно назначить для $SNAPSHOT_Rw права только для чтения
exit;
} fi;
```

Обратите внимание, что в вызове утилиты rsync добавлен ряд исключений. Это сделано для того, чтобы предотвратить синхронизацию всякого мусора, например содержимого кэша веб-браузера, изменяемого слишком часто и занимающего дисковое пространство в каждой копии, потеря которого абсолютно некритична.

Листинг: daily_snapshot_rotate.sh

```
#!/bin/bash
# ————————————————————————————————————————
# Утилита для создания ежедневных мгновенных копий состояния файловой
# системы
# ————————————————————————————————————————
# RCS info: $Id: daily_snapshot_rotate.sh,v 1.2 2002/03/25 21:53:27 mrubel Exp $
# ————————————————————————————————————————
# Этот сценарий предназначен для ежедневного запуска в качестве задания
# планировщика, тогда как мгновенная копия hourly.3 содержит мгновенный
# снимок на 0:00 или, скажем, на 13:00, если предыдущий сценарий
# запускается каждые четыре часа
# ————————————————————————————————————————

# ——- системные команды, используемые сценарием ——————————————
ID=/usr/bin/id;
ECHO=/bin/echo;

MOUNT=/bin/mount;
RM=/bin/rm;
MV=/bin/mv;
CP=/bin/cp;

# ——————————— размещение файлов ————————————————————

MOUNT_DEVICE=/dev/hdb1;
SNAPSHOT_Rw=/root/snapshot;

# ——————————— сам сценарий ————————————————————————

# убедитесь, что вы обладаете правами пользователя root
if (( `$ID -u` != 0 )); then { $ECHO "Sorry, must be root. Exiting..."; exit; } fi

# попытка монтирования файловой системы с правами Rw; в случае неудачи -
# выход
$MOUNT -o remount,rw $MOUNT_DEVICE $SNAPSHOT_Rw ;
if (( $? )); then
{
$ECHO "snapshot: could not remount $SNAPSHOT_Rw readwrite";
# невозможно смонтировать $SNAPSHOT_Rw с правами на чтение и запись
exit;
}
fi;

# этап 1: удаляем самую старую копию, если таковая существует:
if [ -d $SNAPSHOT_Rw/home/daily.2 ] ; then                    \
$RM -rf $SNAPSHOT_Rw/home/daily.2 ;                           \
fi ;

# этап 2: выполняем сдвиг более новых копий на место более старых,
# по одной, если они существуют
if [ -d $SNAPSHOT_Rw/home/daily.1 ] ; then                    \
$MV $SNAPSHOT_Rw/home/daily.1 $SNAPSHOT_Rw/home/daily.2 ;     \
```

```
fi;
if [ -d $SNAPSHOT_Rw/home/daily.0 ]; then                    \
$MV $SNAPSHOT_Rw/home/daily.0 $SNAPSHOT_Rw/home/daily.1;     \
fi;

# этап 3: создаем только жесткие ссылки (за исключением каталогов) для
# копии файловой системы hourly.3, если таковая существует, и копируем ее,
# в мгновенную копию daily.0
if [ -d $SNAPSHOT_Rw/home/hourly.3 ]; then                   \
$CP -al $SNAPSHOT_Rw/home/hourly.3 $SNAPSHOT_Rw/home/daily.0; \
fi;

# внимание: не нужно обновлять значение параметра mtime для daily.0;
# он будет содержать время создания копии hourly.3, что вполне корректно

# сделаем мгновенную копию доступной только для чтения

$MOUNT -o remount,ro $MOUNT_DEVICE $SNAPSHOT_Rw;
if (( $? )); then
{
$ECHO "snapshot: could not remount $SNAPSHOT_Rw readonly";
Невозможно назначить для $SNAPSHOT_Rw права только для чтения
exit;
} fi;
```

Результаты выполнения команды ls -l /snapshot/home:

```
total 28
drwxr-xr-x-12 root root 4096 Mar 28 00:00 daily.0
drwxr-xr-x-12 root root 4096 Mar 27 00:00 daily.1
drwxr-xr-x-12 root root 4096 Mar 26 00:00 daily.2
drwxr-xr-x-12 root root 4096 Mar 28 16:00 hourly.0
drwxr-xr-x-12 root root 4096 Mar 28 12:00 hourly.1
drwxr-xr-x-12 root root 4096 Mar 28 08:00 hourly.2
drwxr-xr-x-12 root root 4096 Mar 28 04:00 hourly.3
```

Обратите внимание: содержимое каждой вложенной директории внутри папки */snapshot/home* представляет собой полный образ папки */home* на момент создания мгновенной копии. Несмотря на наличие атрибута w в файловых разрешениях для данной директории, ни одному пользователю – даже root – не позволено вести запись в эту директорию: она смонтирована только для чтения.

Дополнительная информация:

- *«Синхронизация отдельных частей файловой системы при помощи утилиты rsync»* **[Совет № 41]**;
- веб-сайт *http://www.mikerubel.org/computers/rsync_snapshots/*.

СОВЕТ №43

Работа с ISO и дисками CD-R/CD-RW

(Создание образа стандарта ISO и мгновенных копий из командной строки)

Существует целый ряд утилит под Linux с графическим интерфейсом для работы с CD-ROM. Большинство подобных программ представляют собой лишь клиентский интерфейс, который в свою очередь вызывает утилиты командной строки, выполняющие реальную работу по созданию образа ISO и записи на компакт-диски. Кроме того, графические утилиты не слишком помогут, если к серверу, с которым вы работаете, не подключена консоль, разве что удаленно запущена оболочка X, возможно через ssh (как описывается в разделе *«Запуск X через ssh»* **[Совет № 70]**). Установите на системе утилиты mkisofs и cdrecord, что значительно упростит работу с образами ISO из командной строки.

Аббревиатурой ISO на жаргоне называют файловую систему стандарта ISO9660. Формат ISO9660, в сочетании с рядом расширений, является наиболее распространенным форматом представления данных на CD-дисках на текущий момент.

Для создания образа ISO с целью последующей записи подготовленных данных на компакт-диск воспользуйтесь утилитой mkisofs:

```
mkisofs -r /home/rob > /tmp/rob-home.iso
```

Переключатель -r сообщает о включении расширений Rock Ridge (RR) в результирующем файле образа. Это означает, что при подключении диска к системе, поддерживающей расширения Rock Ridge, будут сохранены длинные имена файлов и разрешения файловой системы. Применяйте переключатель -r, если планируется использовать записанные диски при работе с Linux. Любой пользователь может создавать образы ISO для записи на CD-R из файлов, к которым он имеет доступ. Необязательно обладать правами пользователя root для запуска команды mkisofs. А вот для запуска остальных команд, упоминаемых в данном параграфе, эти привилегии, скорее всего, понадобятся.

Учтите, что команда mkisofs сохраняет содержимое папок, заданных в командной строке, но не сохраняет информацию о самих каталогах. В приведенном выше примере, содержимое директории */home/rob/* будет привязано к /, т.е. помещено в корневой каталог записываемого компакт-диска.

Если необходимо создать образ из уже существующих на CD-диске данных, воспользуйтесь следующей командой:

```
# dd if=/dev/cdrom of=image.iso
```

В результате будет создан образ ISO существующего компакт-диска с данными, объемом в знаменитые 650 Мб, поэтому сначала убедитесь,

что на жестком диске для этого достаточно места. Можно ускорить процесс создания образа, манипулируя параметром bs утилиты dd:

```
# dd if=/dev/cdrom of=image.iso bs=10k
```

Оптимальные значения этого параметра обычно зависят от характеристик жесткого диска и контроллера IDE, поэтому подбирать наилучшие параметры для системы придется эмпирическим путем.

Чтобы подключить созданный образ ISO при помощи утилиты mkisofs или прямиком из dd, выполните следующую команду:

```
# mkdir /mnt/iso
# mount -o loop,ro -t iso9660 ./image.iso /mnt/iso
```

Если после запуска команды mount будет выдано сообщение типа «Could not find any loop device» (Не могу найти устройства обратной связи), это значит, что в системе не установлен соответствующий драйвер устройства обратной связи:

```
# modprobe loop
```

Подключив к системе образ ISO, воспользуйтесь командой cd /mnt/iso и проверьте, какие файлы были помещены в образ. Если с созданным образом будут какие-либо проблемы, воспользуйтесь командой umount /mnt/iso, удалите файл образа и попытайтесь создать его заново.

При записи образа ISO на CD-R(W) воспользуйтесь следующей командой:

```
# cdrecord -v speed=12 dev=0,0,0 -data image.iso
```

Необходимо задать скорость записи для привода (максимальную либо меньше максимальной) в опции speed=. Если перед записью CD-RW диска необходимо его очистить, используйте сначала параметр blank=:

```
# cdrecord -v speed=12 dev=0,0,0 blank=fast -data image.iso
```

Заставить работать пишущий CD-привод в операционной системе Linux не так уж сложно благодаря наличию ide-scsi драйверов. Модуль ядра представляет все так, что CD-приводы IDE (и приводы с другим интерфейсом) воспринимаются системой как обычные SCSI-устройства, которые намного легче программировать.

Дополнительная информация:

- Как заставить работать пишущие CD-приводы в операционной системе Linux см. в разделе CD Writing HOWTO, доступном на сайте документации по проекту Linux, размещенному по адресу *http://www.tldp.org/HOWTO/CD-Writing-HOWTO.html*.

СОВЕТ №44

Запись компакт-дисков без создания файлов образа ISO

(Создание CD путем копирования с другого CD, непосредственно из файловой системы либо из данных, загружаемых из сети интернет)

Самый безопасный способ копирования с диска на диск состоит в создании образа ISO из имеющихся данных и дальнейшей записи на диск готового образа, как было описано в разделе *«Работа с ISO и дисками CD-R/CD-RW»* **[Совет № 43]**. Но иногда сказывается нехватка свободного дискового пространства или времени для выполнения промежуточного шага.

В таком случае, если компьютер достаточно быстрый, можно копировать CD-диски напрямую. Лучше всего это получается, если в системе установлены два CD-привода: один для чтения, другой и для чтения и для записи (как ведущее и ведомое IDE-устройства либо устройства IDE и SCSI).

Чтобы сделать копию CD-диска в режиме реального времени, наберите следующую команду:

```
# dd if=/dev/hdb | cdrecord -v speed=12 dev=0,0,0 fs=8m -data -
```

Аргумент команды cdrecord означает, что трек с данными должен считываться из STDIN, а не из файла. Часть строки с командой dd осуществляет передачу данных команде cdrecord из копии CD, размещенной на вторичном диске, на первичный IDE-привод (hdb). Параметр fs=8m несколько увеличивает размер буфера FIFO, чтобы нивелировать короткие запинки канала передачи данных. Если шина в состоянии справиться с заданием и компьютер не слишком перегружен иными задачами, тогда этот метод будет работать очень хорошо.

Аналогично, при записи на CD-диск файлов с жесткого диска особой нужды создавать образ ISO нет. Попробуйте прибегнуть к следующему способу:

```
# mkisofs -r /home/ftp/ | cdrecord -v speed=12 dev=0,0,0 fs=8m -data -
```

Подобно рассмотренной выше команде dd, команда mkisofs по умолчанию ведет запись в STDOUT. Отсюда данные поступают в STDIN процесса cdrecord, который производит прожиг CD по мере создания образа ISO в режиме реального времени. В таком случае отпадает необходимость в сохранении файла образа ISO на жестком диске. Но будьте осторожны, если данные в источнике занимают объем, больший объема записываемого компакт-диска (больше 650 или 700 Мб) — получите нерабочий диск.

Поэтому, прежде чем начинать запись, проверьте с помощью команды du, сколько понадобится свободного места:

```
# du -hs /home/ftp/
412M /home/ftp
```

Отлично, 412 Мб прекрасно соответствуют требованиям ISO.

Таким образом, любая программа, умеющая передавать данные ISO в STDOUT, является потенциальным кандидатом на роль источника данных для программы прожига. Теперь посмотрим, как обстоят дела с записью CD в режиме реального времени по сети:

```
root@catlin:~# mkisods -r backup/ \
| ssh florian "cdrecord -v speed=12 dev=0,0.0 fs=8m -data -"
```

А как насчет копирования локального CD на удаленно расположенный записывающий привод:

```
root@catlin:~# dd if=/dev/cdrom \
| ssh florian "cdrecord -v speed=12 dev=0,0.0 fs=8m -data -"
```

И что вы думаете о загрузке образа ISO с одновременной его записью:

```
# curl http://my.server.com/slackware-8.1-install.iso \
|cdrecord -v speed=0 dev=0,0,0 fs=8m -data -
```

Мы не будем рекомендовать вам пытаться осуществлять запись по беспроводной сети: для записи компакт-дисков по сети необходим хороший и надежный кабель Ethernet со стабильной пропускной способностью 100 Мбит/с. Да и пример с загрузкой приведен просто в качестве иллюстрации возможностей канала Unix: выходные данные любой команды могут выступать в качестве входных данных практически любой другой команды, благодаря чему можно делать такие вещи, о возможности которых разработчики программ даже не подозревали.

Сеть

Советы № 45-53

Когда-то сетевым администратором называли человека, который тратил все свое время, пытаясь заставить компьютеры выполнять обмен данными друг с другом по сети. Сейчас же складывается впечатление, что сетевые администраторы тратят большую часть времени на решение прямо противоположной задачи — ограничения доступа к компьютерам по сети, отсечения всех нежелательных абонентов и одновременной поддержки легитимного трафика.

Имеющийся в Linux брандмауэр netfilter предоставляет пользователям очень гибкий интерфейс для сетевых решений ядра. При помощи команды iptables для брандмауэра задаются правила, позволяющие создавать богатую и очень гибкую политику доступа. С ее помощью можно проверять соответствие пакетов не только по номеру порта, интерфейсу либо MAC-адресу, но и по данным, содержащимся в пакете, и даже по оценке полученного пакета. Эта информация используется для предупреждения всех возможных видов атак: от переполнения портов до вирусов.

Однако блокировка пользователей все же не так интересна, как их подключение друг к другу. В конце концов, сколько труда было вложено, чтобы позволить пользователям компьютеров общаться друг с другом по сети! В этой главе рассматриваются некоторые не совсем обычные методы контроля сетевого трафика, начиная с удаленного переназначения портов и заканчивая различными формами IP-туннелирования. Изучив инкапсуляцию IP и туннели области пользователя, подобные vtun, вы узнаете, как можно создавать сети, в которых компьютеры связываются друг с другом через интернет и которые предоставляют нетрадиционные, но весьма полезные новые способы поведения.

СОВЕТ №45 Создание брандмауэра из командной строки любого сервера

(Чтобы воспользоваться правилами iptables, не обязательно выделять под брандмауэр отдельный компьютер)

Брандмауэр netfilter (который, начиная с версии Linux 2.4, является встроенным) предоставляет гибкие возможности настройки непосредственно из командной строки. Чтобы научиться работать с iptables, потребуется некоторое время, но зато вскоре вы сможете использовать синтаксис выражений, позволяющий создавать достаточно сложные (и, будем надеяться, полезные) правила брандмауэра.

Даже если компьютер не является брандмауэром — межсетевым экраном — в традиционном смысле этого слова, т.е. обладает единственным сетевым интерфейсом, который не позволяет защищать другие компьютеры, тем не менее, функции подобного фильтра могут быть очень полезны. Предположим, требуется разрешить доступ к компьютеру при помощи службы telnet (только в том случае, если что-то произошло с ssh либо ее библиотеками), но нежелательно, чтобы кто-то другой из просторов всемирной паутины получил доступ к нему. В этом случае воспользуйтесь упаковщиком tcp (tcpwrapper), заполнив файлы */etc/hosts.allow* и */etc/hosts.deny*, а также задав соответствующие настройки в файле */etc/inetd.conf*. Либо вызовите команду iptables с такими параметрами:

```
iptables -A INPUT -t filter -s ! 208.201.239.36 -p tcp --dport 23 -j DROP
```

В большинстве случаев разрешают неограниченный доступ с заслуживающих доверия узлов сети, блокируя доступ с проблемных узлов и устанавливая некоторый промежуточный уровень доступа для всех остальных. Ниже приводится один из методов, позволяющий создавать «белый» и «черный» списки с ограниченной политикой доступа к портам.

```
#
# Простой сценарий инициализации брандмауэра
#
WHITELIST=/usr/local/etc/whitelist.txt
BLACKLIST=/usr/local/etc/blacklist.txt
ALLOWED="22 25 80 443"

#
# Удаляем все существующие правила для фильтров
#
iptables -F

#
# Проанализируем значения переменной $WHITELIST, в которой
# перечислены сети и отдельные узлы с неограниченным обменом данными
#
for x in `grep -v ^# $WHITELIST | awk '{print $1}'`; do
```

```
echo "Permitting $x..."
# Разрешаем...
iptables -A INPUT -t filter -s $x -j ACCEPT
done

#
# Проанализируем "черный список" в переменной $BLACKLIST, заблокировав
# весь трафик с перечисленными в нем сетями и отдельными узлами
#
for x in `grep -v ^# $BLACKLIST | awk '{print $1}'`; do
echo "Blocking $x..."
# Блокируем...
iptables -A INPUT -t filter -s $x -j DROP
done

#
# Для разрешенных портов: какие действия разрешено выполнять
# для узлов сети, не занесенных в "черный список"?
#
for port in $ALLOWED; do
echo "Accepting port $port..."
# Разрешим доступ к портам
iptables -A INPUT -t filter -p tcp --dport $port -j ACCEPT
done

#
# Если иное не оговорено выше, удаляем все входящие запросы
# на установку подключений.
#
iptables -A INPUT -t filter -p tcp —syn -j DROP
```

Не забудьте перечислить все порты, которые хотите включить, в переменной $ALLOWED в начале сценария. Если, например, забудете включить порт № 22, то не сможете запустить службу ssh!

Файл */usr/local/etc/blacklist.txt* содержит список IP-адресов, имен узлов сети и самих сетей:

```
1.2.3.4 # Выполнял сканирование портов 8/15/02
7.8.0.0/24 # Кто знает, какое зло скрывается в засаде
r00tb0y.script-kiddie.coop # $0 s0rR33 u 31337 h4x0r!
```

И, наоборот, в файле */usr/local/etc/whitelist.txt* будет храниться список «хороших парней», которым позволен доступ к системе независимо от других установленных правил:

```
11.22.33.44 # Моя рабочая станция
208.201.239.0/26 # локальная сеть
```

Поскольку система воспринимает строку только до символа #, при желании можете ввести целую строку комментариев. При последующем запуске сценария все строки комментариев будут проигнорированы. Как видите, в начале сценария, перед тем как вводить существую-

щие настройки фильтров, мы запускаем команду iptables -F, так что при внесении изменений в файлы *blacklist.txt* и *whitelist.txt* либо изменении списка портов в параметре $ALLOWED нужно будет просто повторно запустить сценарий.

Учтите также, что этот сценарий подходит только для подключений TCP. Если необходима поддержка UDP, ICMP или каких-то других протоколов, запустите еще один проход, как в случае с переменной $ALLOWED для цикла, но при этом включите дополнительные порты и протоколы (передав в iptables, к примеру, -p -udp или -p icmp).

Будьте осторожны с использованием «белых» списков. Для любой сети или IP-адреса, указанного в этом списке, будет разрешен доступ ко всем портам вашего компьютера. В некоторых обстоятельствах умный злоумышленник может отсылать фальшивые пакеты, пришедшие якобы с указанных адресов, если, конечно, он обладает даром предвидения либо логически вычислил, какие IP-адреса содержатся в вашем списке. Реализация такого типа атаки достаточно сложна, но, тем не менее, возможна. Включайте в список только те используемые во внутренней сети адреса, которые не поддаются трассировке из сети интернет.

Весьма полезно обладать консольным доступом к системе при установке новых правил брандмауэра (вы не сможете заблокировать себе доступ к консоли при помощи *iptables*!). Если вы запутаетесь во время работы с iptables, вспомните о флажке -L — посмотрите с его помощью полный список правил, а затем начните все сначала, выполнив команду iptables -F. Если после ее запуска создается впечатление, что система зависла, попробуйте выполнить команду iptables -L -n, чтобы просмотреть правила, не выполняя разрешения имен DNS: возможно, установленные правила привели к блокировке DNS.

Даже при помощи обычной фильтрации можно сделать немало, однако, не забывайте, что функциональные возможности iptables не ограничиваются простой фильтрацией запросов, пришедших с целевых систем.

Дополнительная информация:

- раздел «Как работать с брандмауэром Netfilter» (HOWTO) по адресу *www.netfilter.org*;
- *«Советы и уловки по работе с iptables»* **[Совет № 47]**.

СОВЕТ **№46** **Простая подмена IP-адреса**
(Настраиваем NAT — транслятор сетевых адресов)

Чтобы организовать общий доступ к сети интернет через один IP-адрес из локальной сети, можете воспользоваться функцией подмены IP-

адреса на узле сети, выступающем в роли шлюза. При помощи iptables это реализуется в две строки:

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -A POSTROUTING -o $EXT_IFACE -j MASQUERADE
```

где $EXT_IFACE представляет собой внешний интерфейс шлюза. Теперь любой компьютер, находящийся внутри локальной сети, либо любой другой интерфейс шлюза сможет подключаться к сети интернет через ваш компьютер. Причем во всех запросах, отправляемых во всемирную паутину через шлюз, в качестве IP-адреса отправителя будет значиться внешний IP-адрес шлюза.

Вот тут нужно «позаботиться» о злоумышленниках из внешних сетей, отсылающих на ваш шлюз пакеты с фальсифицированными IP-адресами, якобы принадлежащими клиентам внутренней сети. Такие пакеты позволяют ввести в заблуждение даже системное ядро, которое воспримет этот трафик как вполне законный. В результате, злоумышленник, выдающий себя за клиента внутренней сети, может организовать атаку, которая будет символизировать наступление тяжелых времен для бесталанного владельца шлюза.

Надеемся, что данная проблема останется в прошлом, поскольку в последние версии ядра специально для решения этой проблемы включено дополнительное правило брандмауэра под названием rp_filter. Его действие заключается в том, что при поступлении на интерфейс пакета, отправитель которого не соответствует записям таблицы маршрутизации, пакет удаляется. Это эффективное средство предотвращения фальсификации IP-адресов, разрешающее выполнение простой и безопасной подмены в вышеприведенном примере.

В тех очень маловероятных случаях, когда использование rp_filter вызывает какие-то проблемы, его можно легко деактивировать:

```
# echo "0" /proc/sys/net/ipv4/conf/all/rp_filter
```

Дополнительная информация:

- *«Советы и уловки по работе с iptables»* [Совет № 47].

СОВЕТ №47

Советы и хитрости в работе с iptables

(Заставьте свой брандмауэр выполнять более сложные задачи, чем фильтрация пакетов с помощью iptables)

Утилита iptables представляет собой новое поколение программных брандмауэров в рамках проекта netfilter. Она включает в себя функциональность всех своих предшественников, а также, помимо

брандмауэров с запоминанием состояния (stateful firewall), поддерживает цепочки ip (ipchains). Утилита также имеет базу для расширения функциональных возможностей при помощи загружаемых модулей. Кроме того, при работе с базовой версией iptables можете воспользоваться некоторыми уловками, а также рядом модулей для ее расширения.

Для выполнения следующего примера необходимо настроить такие переменные среды:

$EXT_IFACE

Внешний (открытый) интерфейс брандмауэра.

$INT_IFACE

Внутренний (закрытый) интерфейс брандмауэра.

$DEST_IP

Полный запрошенный получатель пакета.

Вы можете применять iptables для ограничения входящих TCP-пакетов с целью предотвращения атак по типу «отказа от обслуживания». Решение этой проблемы проработано в следующих правилах:

```
# Создание цепочки syn-flood для обнаружения атак по
# типу "отказа от обслуживания"
iptables -t nat -N syn-flood

# Задание ограничения в 12 подключений в секунду
# (с пиковым значением в 24)
iptables -t nat -A syn-flood -m limit --limit 12/s --limit-burst 24\
   -j RETURN
iptables -t nat -A syn-flood -j DROP

# Выявление атаки по типу "отказа от обслуживания"
iptables -t nat -A PREROUTING -I $EXT_IFACE -d $DEST_IP -p tcp --syn \
   -j syn-flood
```

Эти правила позволяют ограничить установку новых входящих подключений TCP (с битом SYN) до 12 подключений в секунду после достижения пикового значения в 24 новых подключения в секунду.

При помощи iptables можно достаточно прозрачно настроить прокси Squid. Этот прокси-сервер помогает кэшировать и записывать в журнал все исходящие запросы HTTP, направленные в интернет. Он не требует внесения каких-либо модификаций в браузер пользователя и подходит для отсева нежелательного содержимого. Эта проблема решена при помощи следующего правила iptables наверху цепочки PREROUTING:

```
#Прозрачная настройка прокси-сервера Squid для внутренней сети
#
#Более подробную информацию по настройкам прокси-сервера Squid вы найдете
#по адресу: http://www.linuxdoc.org/HOWTO/mini/TransparentProxy.html
#
iptables -t nat -A PREROUTING -i $INT_IFACE -p tcp --dport 80 /
-j REDIRECT --to-port 3128
```

Это правило осуществляет перенаправление исходящих запросов по 80-му порту TCP на прокси-сервер Squid, работающий через брандмауэр по TCP-порту 3128.

С помощью iptables можно назначать соответствие для произвольных флагов TCP. Это означает возможность блокировки дерева XMAS (когда установлены все флаги) и NULL-пакетов при помощи следующих правил:

```
#Удаление TCP-пакетов XMAS и NULL
iptables -t nat -A PREROUTING -p --tcp-flags ALL ALL -j DROP
iptables -t nat -A PREROUTING -p --tcp-flags ALL NONE -j DROP
```

Дополнительные возможности iptables

Утилита iptables предоставляет ряд дополнительных функций брандмауэра, реализуемых путем установки «заплат» для ядра Linux. Эти «заплаты» можно загрузить с сайта *http://www.netfilter.org/*. Здесь размещены специальные «заплаты» типа patch-o-matic (набор «заплат» для исходных текстов ядра, выпускаемых вместе с комплектом сценариев, которые выполняют их установку, что делает возможным работу с POM даже для новичков). Необходимо выбирать ту версию, которая в точности соответствует используемой версии утилиты iptables. Эти «заплаты» пока что недоступны в общих поставках ядра Linux. Некоторые из них являются экспериментальными, поэтому к их установке и использованию следует относиться с осторожностью.

При помощи экспериментальной «заплаты» psd для netfilter утилита iptables может обнаруживать и блокировать случаи входящего сканирования портов. Для этого потребуется задать такое правило:

```
#Предотвращение входящего сканирования портов
iptables -t nat -A PREROUTING -i $EXT_IFACE -d $DEST_IP -m psd -j DROP
```

При помощи другой экспериментальной «заплаты» iplimit утилита iptables может ограничивать количество подключений с определенного IP-адреса путем задания такого правила:

```
#Игнорирование пакетов, приходящих с узла сети,
#с которым установлено
#более 16 активных подключений
```

```
iptables -t nat -A PREROUTING -i $EXT_IFACE -p tcp --syn -d $DEST_IP -m
iplimit --iplimit-above 16 -j DROP
```

Одна из наиболее мощных «заплат» для netfilter помогает дифференцировать пакеты по их содержимому. Экспериментальная »заплата», предназначенная для поиска строки соответствия, позволяет отсеивать пакеты, в которых содержится некоторая строка символов. Эта функция дает возможность выявлять на ранних стадиях вирусы CodeRed и Nimda, до того как они успеют поразить веб-сервер. Следующие правила позволяют достичь того же эффекта:

```
# Удаление пакетов HTTP, связанных с вирусами CodeRed и Nimda без запроса
iptables -t filter -A INPUT -i $EXT_IFACE -p tcp -d $DEST_IP --dport http \
-m string --string "/default.ida?" -j DROP
iptables -t filter -A INPUT -i $EXT_IFACE -p tcp -d $DEST_IP --dport http \
-m string --string ".exe?/c+dir" -j DROP
iptables -t filter -A INPUT -i $EXT_IFACE -p tcp -d $DEST_IP --dport http \
-m string --string ".exe?/c+tftp" -j DROP
```

Теперь в iptables встроена поддержка функции переназначения портов. Для реализации этой возможности таблица трансляции сетевых адресов использует функцию Destination NAT (транслятор сетевых адресов получателя) в цепочке PREROUTING. Чтобы перенаправить обращения HTTP *на узел 10.0.0.3 внутренней сети, можно написать* такое правило:

```
# Использование DNAT для переназначения порта http
iptables -t nat -A PREROUTING ! -i $INT_IFACE -p tcp --destination-port \
80 -j DNAT --to 10.0.0.3:80
```

Можно также переназначать порты для пакетов UDP. Если перенаправить трафик для определенного порта, то не придется создавать соответствующее правило в цепочке INPUT для приема входящих подключений к данному порту. Это сработает только в том случае, если в качестве получателя будет выступать сеть с локально подключенным интерфейсом, т.е. получателем не может являться чужая сеть. Советуем обратить внимание на утилиты типа rinetd (см. раздел *«Перенаправление трафика через порты TCP на произвольные узлы сети»* **[Совет № 48]**) или nportredird, если требуется перенаправить трафик на удаленные сети.

При выполнении перенаправления запросов HTTP на узлы внутренней сети можете также включить проверку трафика на вирус CodeRed в цепочке команд FORWARD:

```
iptables -t filter -A FORWARD -p tcp --dport http \
-m string --string "/default.ida?" -j DROP
```

На первых порах использование утилиты iptables может вызывать некоторые сложности, однако гибкость делает ее чрезвычайно полез-

ной. Если вы запутаетесь во время написания правил (а это рано или поздно обязательно случится), помните, что вашими лучшими друзьями являются команды iptables -L -n и tcpdump, возможно, сопровождаемые быстрой установкой сессии с помощью ethereal.

Дополнительная информация:

- *«Перенаправление трафика через порты TCP на произвольные узлы сети»* **[Совет № 48]**;
- раздел HOWTO для брандмауэра Linux.

СОВЕТ №48 Перенаправление трафика через порты TCP на произвольные узлы сети

(Заставьте систему воспринимать удаленные службы как локальные)

Как было показано в разделе *«Советы и уловки при работе с iptables»* **[Совет № 47]**, можно легко перенаправлять трафик, поступающий на порты TCP и UDP, с брандмауэра на внутренние узлы сети при помощи утилиты iptables. Но что, если понадобится перенаправить трафик с произвольного узла сети на систему, которая находится за пределами вашей сети? В таком случае следует воспользоваться программой переназначения портов, работающей на уровне приложений, типа rinetd.

Эта утилита далеко не нова, однако она выигрывает за счет своего малого размера, эффективности и потому идеально подходит для решения подобных проблем. Для ее установки достаточно распаковать архив и запустить утилиту make, чтобы получить небольшой по размеру исполняемый файл, позволяющий перенаправлять трафик через порты TCP. К сожалению, в rinetd не поддерживается работа с портами UDP.

Конфигурационный файл выглядит до неприличия просто:

```
[Адрес источника] [Порт источника] [Адрес получателя] [Порт получателя]
```

Каждый порт, трафик с которого нуждается в перенаправлении, должен задаваться на отдельной строке. В качестве адреса источника или получателя могут выступать имена узлов сети и IP-адреса, причем IP-адрес 0.0.0.0 связывает rinetd с любым доступным локальным IP:

```
0.0.0.80 some.othersite.gov 80
216.218.203.211 25 123.45.67.89.25
0.0.0.0 5353 my.shellserver.us 22
```

Сохраните эти строки в файл под именем */ect/rinetd.conf*, а затем скопируйте утилиту rinetd, например, в папку */usr/local/sbin/*. Для ее запуска надо просто набрать в командной строке **rinetd**.

Первая строка примера выполняет перенаправление всего веб-трафика, поступающего на локальные адреса на веб-сервер *some.other-site.gov*. Обратите внимание: это будет работать только в том случае, если локальный порт 80 не занят другим процессом, например Apache.

Во второй строке осуществляется перенаправление входящего трафика SMTP, поступившего на узел 216.218.203.211 на почтовый сервер 123.45.67.89 (не влияя при этом на все остальные агенты SMTP, привязанные к локальным адресам). И в последней строке примера весь входящий трафик по порту 5353 перенаправляется на сервер ssh, размещенный по адресу *my.shellserver.us*. Все это работает без транслятора сетевых адресов (NAT) и каких-либо специальных настроек ядра. Просто запустите rinetd, и он превратится в активного демона, занятого прослушиванием заданных вами портов.

Когда возникает необходимость изменения адресов для них либо их физического перемещения эта утилита позволяет значительно облегчить переход на новые серверы, сохраняя для служб впечатление запуска с тех же самых IP-адресов, даже если теперь они размещены вообще в другой сети. Если изменяются правила для портов с номерами больше 1024, то для запуска службы rinetd не обязательно обладать правами пользователя root. Кроме того, эта служба обладает широкими возможностями в плане обеспечения контроля доступа и сохранения журналов. Эту небольшую утилиту полезно иметь в тех случаях, когда возникает необходимость в непрямом доступе к портам TCP.

Дополнительная информация:

- *http://www.boutell.com/rinetd/*;
- *«Советы и уловки по работе с iptables»* **[Совет № 47]**.

СОВЕТ №49

Применение пользовательских цепочек iptables

(Контролируйте брандмауэр с помощью пользовательских цепочек)

По умолчанию, таблица фильтров утилиты iptables включает в себя три цепочки: INPUT, FORWARD и OUTPUT. Но вы можете добавлять столько пользовательских цепочек, сколько посчитаете нужным, чтобы упростить управление большими наборами правил. Пользовательские цепочки ведут себя подобно встроенным, обеспечивая логику, которая должна быть передана до того, как будет определена окончательная судьба пакета.

Для создания новой цепочки воспользуйтесь переключателем -N:

```
root@mouse:~# iptables -N fun-filter
```

Посмотреть, какие цепочки определены в данный момент, можно при помощи стандартного переключателя -L:

```
root@mouse:~# iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination

Chain fun-filter (0 references)
target prot opt source destination
```

Чтобы воспользоваться созданной пользовательской цепочкой, необходимо перейти к ней, сделав «прыжок». Добавим переход к только что созданной цепочке fun-filter непосредственно из цепочки INPUT:

```
root@mouse:~# iptables -t filter -A INPUT -j fun-filter
```

Теперь пользовательская цепочка может достичь любого желаемого вида сложности. Например, можете задать соответствие пакетов на основе MAC-адресов:

```
root@mouse:~# iptables -A fun-filter -m mac --mac-source 11:22:33:aa:bb:cc \
-j ACCEPT
root@mouse:~# iptables -A fun-filter -m mac --mac-source de:ad:be:ef:00:42 \
-j ACCEPT
root@mouse:~# iptables -A fun-filter -m mac --mac-source 00:22:44:fa:ca:de
-j REJECT --reject-with icmp-host-unreachable
root@mouse:~# iptables -A fun-filter -j RETURN
```

Команда RETURN в конце таблицы завершает обработку пакетов и осуществляет возврат к вызывавшей ее цепочке (в нашем случае, цепочке INPUT). Просмотрим, как выглядят наши таблицы, при помощи переключателя -L:

```
root@mouse:~# iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination
fun-filter all -- anywhere anywhere

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination

Chain fun-filter (0 references)
target prot opt source destination
```

```
ACCEPT all -- anywhere anywhere MAC 11:22:33:AA:BB:CC
ACCEPT all -- anywhere anywhere MAC DE:AD:BE:EF:00:42
ACCEPT all -- anywhere anywhere MAC 00:22:44:FA:CA:DE reject-with
icmp-host-unreachable
RETURN all -- anywhere anywhere
```

Можно осуществлять переход к любым предварительно определенным пользовательским цепочкам и даже переключаться между ними. Это помогает изолировать правила, находящиеся в процессе тестирования, от стандартных правил системной политики, а также легко отключать и подключать их снова. Если захотите временно приостановить использование пользовательской цепочки, можете просто удалить команду перехода из цепочки INPUT, вместо того чтобы удалять всю пользовательскую цепочку:

```
root@mouse:~# iptables -t filter -D INPUT -j fun-filter
```

Если решите удалить свою пользовательскую цепочку, воспользуйтесь переключателем -X:

```
root@mouse:~# iptables -X fun-filter
```

Учтите, что возможна ситуация, когда ссылки на локальные цепочки отсутствуют, что обнаруживается при попытке их удаления. В таком случае воспользуйтесь переключателем -F для удаления, в первую очередь, самой цепочки, если присутствуют правила, ссылающиеся на нее.

При надлежащем управлении даже самые сложные правила утилиты iptables будут просты для чтения, если воспользоваться интуитивно понятным наименованием пользовательских цепочек.

Дополнительная информация:

- *«Советы и хитрости в работе с iptables»* **[Совет № 47]**.

СОВЕТ №50

Туннелирование: инкапсуляция IPIP

(IP-туннелирование при помощи драйвера IPIP в Linux)

Если вы раньше никогда не занимались туннелированием IP, *желательно обратиться к руководству «Advanced Router HOWTO»*, прежде чем продолжать чтение данной главы. По сути, IP-туннелирование во многом напоминает использование виртуальных частных сетей (VPN), с тем лишь отличием, что не в каждом туннеле IP используется шифрование. Компьютер, связанный с другой сетью при помощи туннеля, имеет виртуальный интерфейс не с локальным IP-адресом, а с IP-адресом, существующим где-то в удаленной сети. Обычно весь сетевой трафик направляется в этот туннель, так что

удаленные клиенты оказываются доступными для сетевых служб, либо, в более общем смысле, можно соединить две частные сети между собой, используя для передачи туннелированного трафика сеть интернет.

Если необходимо организовать простейший туннель IP между двумя компьютерами, воспользуйтесь на практике туннелированием IPIP. Вероятно, это простейший протокол туннелирования, который поддерживается в операционных системах *BSD, Solaris и даже Windows. Учтите, что IPIP — это всего лишь протокол туннелирования, и поэтому он не включает в себя какую-либо разновидность шифрования. Кроме того, он умеет осуществлять туннелирование только при обмене пакетами между двумя точками; если вам необходимо осуществлять туннелирование широковещательного трафика, обратитесь к разделу *«Туннелирование: инкапсуляция GRE»* **[Совет № 51]**.

Перед созданием туннеля необходимо скопировать расширенный набор утилит по маршрутизации (особенно утилиту ip). Самую последнюю официальную версию можно загрузить из сети интернет по адресу: *ftp://ftp/inr.ac.ru/ip-routing/*. Предупреждаем, что дополнительные утилиты для маршрутизации не отличаются дружественностью по отношению к пользователю, но зато они позволяют манипулировать практически любой стороной работы сетевого модуля Linux.

Предположим, что имеются две частные сети (10.42.1.0/24 и 10.42.2.0/24), причем обе они подключены к интернету напрямую через маршрутизаторы Linux. «Реальный» IP-адрес маршрутизатора первой сети — 240.101.83.2, маршрутизатора второй сети — 251.4.92.217.

Во-первых, загрузим модули ядра на обоих маршрутизаторах:

```
# modprobe ipip
```

Затем на маршрутизаторе первой сети (10.42.1.0/24) запустим следующую команду:

```
# ip tunnel add mytun mode ipip remote 251.4.92.217 \
 local 240.101.83.2 ttl 255
# ifconfig mytun 10.42.1.1
# route add -net 10.42.2.0/24 dev mytun
```

После чего на маршрутизаторе второй сети (10.42.2.0/24) запустим обратную команду:

```
# ip tunnel add mytun mode ipip remote 240.101.83.2 \
 local 251.4.92.217 ttl 255
# ifconfig 10.42.2.1
# route add -net 10.42.1.0/24 dev mytun tun10
```

Разумеется, можете присвоить интерфейсу и более значимое имя, чем mytun. Теперь с первого маршрутизатора должны проходить запросы команды ping на адрес 10.42.2.1, а с маршрутизатора второй сети — запросы команды ping на адрес 10.42.1.1. То есть, любой компьютер, принадлежащий сети 10.42.1.0/24, должен иметь возможность обращаться к любому компьютеру сети 10.42.2.0/24 и наоборот так, будто сеть интернет тут вообще ни при чем.

Если на системе запущено ядро Linux версий 2.2x, в нем существует ярлык, который можно использовать вместо пакета утилит Advanced Router (расширенный маршрутизатор). После загрузки модуля попробуйте запустить следующие команды:

```
# ifconfig tun10 10.42.1.1 pointopoint 251.4.92.217
# route add -net 10.42.2.0/24 dev tun10
```

И на маршрутизаторе второй сети (10.42.2.0/24):

```
# ifconfig tun10 10.42.2.1 pointopoint 240.101.83.2
# route add -net 10.42.1.0/24 dev tun10
```

Если получен отклик на запросы команды ping от противоположного маршрутизатора, однако создалось впечатление, что остальные компьютеры в сети не в состоянии передавать трафик через маршрутизатор, убедитесь, что оба маршрутизатора настроены на пересылку пакетов между интерфейсами:

```
# echo "1" > /proc/sys/net/ipv4/ip_forward
```

Чтобы получить доступ к еще одной сети, помимо 10.42.1.0 и 10.42.2.0, достаточно добавить дополнительные строки route add -net... Не потребуется задавать дополнительные настройки для других узлов сети, если они используют стандартный маршрут передачи данных к соответствующему маршрутизатору (что они определенно должны делать, это же их маршрутизатор).

Для отключения туннеля нужно: отключить интерфейсы на обоих маршрутизаторах и при желании удалить их:

```
# ifconfig mytun down
# ip tunnel del mytun
```

Либо, в Linux 2.2:

```
# ifconfig tun10 down
```

Ядро осторожно очистит таблицу маршрутизации, после того как интерфейсы будут удалены.

Дополнительная информация:

- статья, посвященная расширенным возможностям маршрутизации, по адресу *http://www.tldp.org/ HOWTO/Adv-Routing-HOWTO/*;
- информация по утилитам для расширенной маршрутизации (iproute2), *ftp://ftp.inr.ac.ru/ip-routing/*.

СОВЕТ №51

Туннелирование: инкапсуляция GRE

(IP-туннели с инкапсуляцией общей маршрутизации (GRE))

Аббревиатура GRE расшифровывается как Generic Routing Encapsulation — инкапсуляция общей маршрутизации. Подобно туннелированию IPIP (см. раздел *«Туннелирование: инкапсуляция IPIP»* **[Совет № 50]**), трафик передается через туннель GRE в незашифрованном виде. Основное преимущество GRE по сравнению с IPIP заключается в возможности широковещательной передачи пакетов, к тому же данный вид туннелирования совместим с маршрутизаторами Cisco.

Аналогично примеру, описывающему туннелирование IPIP, рассмотрим ситуацию с двумя сетями (10.42.1.0/24 и 10.42.2.0/24), напрямую связанными с интернетом через маршрутизатор Linux. Причем реальный IP-адрес маршрутизатора первой сети – 240.101.83.2, а второй – 251.4.92.217.

Как и в только что рассмотренном разделе «Туннелирование: инкапсуляция IPIP», нам понадобится копия пакета расширенных утилит маршрутизации. После установки пакета iproute2 можно приступать к загрузке модуля GRE на обоих маршрутизаторах:

```
# modprobe ip_gre
```

Создайте новое «устройство» (dev) туннеля на маршрутизаторе первой сети:

```
# ip tunnel add gre0 mode gre remote 251.4.92.217 local 240.101.83.2 ttl 255
# ip addr add 10.42.1.254 dev gre0
# ip link set gre0 up
```

Обратите внимание: можете назвать «устройством» практически все, что угодно, и gre0 – лишь один из возможных примеров. Точно так же вместо адреса 10.42.1.254 можете подставить любой доступный IP-адрес первой сети (кроме 10.42.1.1, связаного с внутренним интерфейсом). Теперь задайте сетевые маршруты через новый интерфейс туннеля:

```
# ip route add 10.42.2.0/24 dev gre0
```

Переходим ко второй сети:

```
# ip tunnel add gre0 mode gre remote 240.101.83.2 local 251.4.92.217 ttl 255
# ip addr add 10.42.2.254 dev gre0
# ip link set gre0 up
# ip route add 10.42.1.0/24 dev gre0
```

И снова вместо адреса 10.42.2.254 можете подставить любой доступный IP-адрес второй сети. Можете свободно добавлять столько команд ip route add ... dev gre0, сколько считаете необходимым.

Теперь пересылка пакетов между двумя сетями происходит как бы без участия интернета.

Запуск команды traceroute выдаст информацию всего о нескольких переходах к любому узлу второй сети, хотя, если связь не идеальна, можно заметить задержку при переходе к узлу 10.42.2.254. Если возникли проблемы, прочтите примечания к примерам с туннелированием IPIP. Лучшим помощником в этом случае станет сетевой монитор типа tcpdump или ethereal. Чтобы увидеть полную картину происходящего, во время выполнения утилиты ping запустите на обоих маршрутизаторах команду tcpdump 'proto \icmp'.

Для отключения туннеля требуется запустить на обеих системах команду:

```
# ip link set gre0 down
# ip tunnel del gre0
```

Дополнительная информация:

- статья, посвященная расширенным возможностям маршрутизации, по адресу *http://www.tldp.org/ HOWTO/Adv-Routing-HOWTO/*;
- сведения по утилитам для расширенной маршрутизации (iproute2), *ftp://ftp.inr.ac.ru/ip-routing/*.

СОВЕТ №52 Использование vtun через ssh для обмана транслятора сетевых адресов

(Соедините две сети с помощью vtun и единственного подключения ssh)

Утилита vtun представляет собой сервер туннеля области пользователя, позволяющий реализовывать туннелирование целых сетей с помощью универсального драйвера туннеля tun ядра Linux. Подключения могут создаваться непосредственно через IP, PPP и даже через последовательный порт. Данная технология подходит для случаев ограничения общего сетевого доступа вмешательством брандмауэра, когда весь трафик IP шифруется и перенаправляется через единственный порт TCP, т.е. через единственное подключение ssh.

Описанная ниже процедура позволяет узлам сети, имеющим частный IP-адрес (10.42.4.6), создавать новый интерфейс туннеля с реальным трассируемым IP-адресом (208.201.239.33), который работает, как будто бы никакой частной сети не существует в помине. Это можно реализовать созданием нового туннеля, удалив старый маршрут по умолчанию, а затем добавив новый маршрут по умолчанию через другой конец туннеля.

Конфигурация сети до создания туннеля:

```
root@client:~# ifconfig eth2
eth2 Link encap:Ethernet HWaddr 00:02:2D:2A:27:EA
inet addr:10.42.3.2 Bcast:10.42.3.63 Mask:255.255.255.192
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:662 errors:0 dropped:0 overruns:0 frame:0
TX packets:733 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:100
RX bytes:105616 (103.1 Kb) TX bytes:74259 (72.5 Kb)
Interrupt:3 Base address:0x100

root@client:~# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
10.42.3.0 * 255.255.255.192 U 0 0 0 eth2
loopback * 255.0.0.0 U 0 0 0 lo
default 10.42.3.1 0.0.0.0 UG 0 0 0 eth2
```

Как видите, нашей локальной сети принадлежат адреса 10.42.3.0/26, IP-адрес нашего узла — 10.42.3.2, а шлюз по умолчанию имеет адрес 10.42.3.1. Шлюз обеспечивает трансляцию сетевых адресов для сети интернет. Вот так, приблизительно, будет выглядеть наш путь к сайту yahoo.com:

```
root@client:~# traceroute -n -yahoo.com
traceroute to yahoo.com (64.58.79.230), 30 hops max, 40 byte packets
1 10.42.3.1 2.848 ms 2.304 ms 2.915 ms
2 209.204.179.1 16.654 ms 16.052 ms 19.224 ms
3 208.201.224.194 20.112 ms 20.863 ms 18.238 ms
4 208.201.224.5 213.466 ms 338.259 ms 357.7 ms
5 206.24.221.217 20.743 ms 23.504 ms 24.192 ms
6 206.24.210.62 22.379 ms 30.948 ms 54.475 ms
7 206.24.226.104 94.263 ms 94.192 ms 91.825 ms
8 206.24.238.61 97.107 ms 91.005 ms 91.133 ms
9 206.24.238.26 95.443 ms 98.846 ms 100.055 ms
10 216.109.66.7 92.133 ms 97.419 ms 94.22 ms
11 216.33.98.19 99.491 ms 94.661 ms 100.002 ms
12 216.35.210.126 97.945 ms 93.608 ms 95.347 ms
13 64.58.77.41 98.607 ms 99.588 ms 97.816 ms
```

В этом примере мы подключаемся к серверу туннеля в интернете с IP-адресом 208.201.239.5. Для него назначены два свободных IP-ад-

реса (208.201.239.32 и 208.201.239.33), которые могут использоваться в целях туннелирования. Назовем этот компьютер *сервером*, а нашу локальную систему — *клиентом*.

Теперь заставим наш туннель работать. Для начала загрузим драйвер tun на обоих компьютерах:

```
# modprobe tun
```

Если версии ядра сервера и клиента не совпадают, этот драйвер может давать сбои. Для обеспечения бесперебойной работы рекомендуем использовать последние версии ядра, причем одинаковые, например, 2.4.19, на обоих компьютерах.

Напишите следующий программный код и сохраните его на сервере в файл */usr/local/etc/vtund.conf*:

```
options {
port 5000;
ifconfig /sbin/ifconfig;
route /sbin/route;
syslog auth;
}

default {
compress no;
speed 0;
}

home {
type tun;
proto tcp;
stat yes;
keepalive yes;

pass sHHH; # ТРЕБУЕТСЯ пароль.

up {
ifconfig "%% 208.201.239.32 pointopoint 208.201.239.33";

program /sbin/arp "-Ds 208.201.239.33 %% pub";
program /sbin/arp "-Ds 208.201.239.33 eth0 pub";

route "add -net 10.42.0.0/16 gw 208.201.239.33";
};

down {
program /sbin/arp "-d 208.201.239.33 -i %%";
program /sbin/arp "-d 208.201.239.33 -i eth0";

route "del -net 10.42.0.0/16 gw 208.201.239.33";
};
}
```

Запустите сервер vtund при помощи команды:

```
root@server:~# vtund -s
```

Создайте файл *vtund.conf* для клиента. Наберите следующий код на системе клиента и сохраните его в файл с именем */usr/local/etc/vtund.conf*:

```
options {
port 5000;
ifconfig /sbin/ifconfig;
route /sbin/route;
}

default {
compress no;
speed 0;
}

home {
type tun;
proto tcp;
keepalive yes;

pass sHHH; # ТРЕБУЕТСЯ пароль.

up {
ifconfig "%% 208.201.239.33 pointopoint 208.201.239.32 arp";

route "add 208.201.239.5 gw 10.42.3.1";
route "del default";
route "add default gw 208.201.239.32";

};

down {
route "del default";
route "del 208.201.239.5 gw 10.42.3.1";
route "add default gw 10.42.3.1";
};
}
```

И, наконец, запустите на системе клиента следующую команду:

```
root@client:~# vtund -p home server
```

Теперь в нашем распоряжении имеется не только туннель между клиентом и сервером, но и новый маршрут по умолчанию, установленный с другого конца туннеля. Посмотрим, что произойдет, когда мы обратимся к веб-сайту yahoo.com через активный туннель с помощью утилиты traceroute:

```
root@client:~# traceroute -n -yahoo.com
traceroute to yahoo.com (64.58.79.230), 30 hops max, 40 byte packets
1 208.201.239.32 24.368 ms 28.019 ms 19.114 ms
2 208.201.239.1 21.677 ms 22.644 ms 23.489 ms
3 208.201.224.194 20.41 ms 22.997 ms 23.788 ms
4 208.201.224.5 26.496 ms 23.8 ms 25.752 ms
5 206.24.221.217 26.174 ms 28.077 ms 26.344 ms
6 206.24.210.62 26.484 ms 27.851 ms 25.015 ms
7 206.24.226.103 104.22 ms 114.278 ms 108.575 ms
8 206.24.238.57 99.978 ms 99.028 ms 100.976 ms
9 206.24.238.26 103.749 ms 101.416 ms 101.09 ms
10 216.109.66.132 102.426 ms 104.222 ms 98.675 ms
11 216.33.98.19 99.985 ms 99.618 ms 103.827 ms
12 216.35.210.126 104.075 ms 103.247 ms 106.398 ms
13 64.58.77.41 107.219 ms 106.285 ms 101.169 ms
```

Это означает, что любой серверный процесс, запущенный на клиенте, теперь может быть доступен из интернета по адресу 208.201.239.33. И все это возможно без внесения каких-либо изменений на шлюзе 10.42.3.1, например переназначения портов.

Вот как будет выглядеть новый интерфейс туннеля на клиенте:

```
root@client:~# ifconfig tun0
tun0 Link encap:Ethernet Point-to-Point Protocol
inet addr:208.201.239.33 P-t-P:208.201.239.32 Mask:255.255.255.255
UP POINTOPOINT RUNNING MULTICAST MTU:1500 Metric:1
RX packets:39 errors:0 dropped:0 overruns:0 frame:0
TX packets:39 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:10
RX bytes:2220 (2.1 Kb) TX bytes:1560 (1.5 Kb)
```

Здесь же приводится обновленная таблица маршрутизации. Учтите, что по-прежнему необходимо хранить маршрут от узла сети до IP-адреса сервера туннеля через наш старый шлюз по умолчанию в противном случае трафик нельзя будет направить через туннель:

```
root@client:~# route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
208.201.239.5 10.42.3.1 255.255.255.255 UGH 0 0 0 eth2
208.201.239.32 * 255.255.255.255 UH 0 0 0 tun0
10.42.3.0 * 255.255.255.192 U 0 0 0 eth2
10.42.4.0 * 255.255.255.192 U 0 0 0 eth0
loopback * 255.0.0.0 U 0 0 0 lo
default 208.201.239.32 0.0.0.0 UG 0 0 0 tun0
```

Для отключения туннеля достаточно применить команду kill к процессу vtund на системе клиента. В результате все сетевые настройки будут возвращены в исходное состояние.

Данный метод хорошо работает, если вы доверяете vtun при условии использования устойчивого шифрования и при этом защищены от уязвимостей, доступных для удаленного использования. Чтобы воспользоваться vtun через ssh (со строгой аутентификацией и шифрованием, обеспечиваемым ssh), переназначьте порт 5000 системы клиента на тот же самый порт сервера. Это будет выглядеть примерно так:

```
root@client:~# ssh -f -N -c blowfish -C -L5000:localhost:5000 server
root@client:~# vtund -p home localhost
root@client:~# traceroute -n yahoo.com
traceroute to yahoo.com (64.58.79.230), 30 hops max, 40 byte packets
1 208.201.239.32 24.715 ms 31.713 ms 29.519 ms
2 208.201.239.1 28.389 ms 36.247 ms 28.879 ms
3 208.201.224.194 48.777 ms 28.602 ms 44.024 ms
4 208.201.224.5 38.788 ms 35.608 ms 35.72 ms
5 206.24.221.217 37.729 ms 38.821 ms 43.489 ms
6 206.24.210.62 39.577 ms 43.784 ms 34.711 ms
7 206.24.226.103 110.761 ms 111.246 ms 117.15 ms
8 206.24.238.57 112.569 ms 113.2 ms 111.773 ms
9 206.24.238.26 111.466 ms 123.051 ms 118.58 ms
10 216.109.66.132 113.79 ms 119.143 ms 109.934 ms
11 216.33.98.19 111.948 ms 117.959 ms 122.269 ms
12 216.35.210.126 113.472 ms 111.129 ms 118.079 ms
13 64.58.77.41 110.923 ms 110.733 ms 115.22 ms
```

Чтобы не допустить подключений из внешнего мира к vtund через порт 5000 сервера, добавьте в брандмауэр netfilter следующее правило:

```
root@server:~# iptables -A INPUT -t filter -i eth0 -p tcp --dport 5000 -j DROP
```

Таким образом, разрешается установка локальных подключений, поскольку они используют интерфейс обратной связи и требуют создания туннеля ssh с сервером до установления подключения.

Помимо присвоения реальных IP-адресов компьютерам, находящимся за пределами действия транслятора сетевых адресов, можно эффективно связывать вместе две любые сети, установив между ними единственное подключение ssh, инициированное с любого конца.

Если вы плохо сориентировались в настройках *vtund.conf* и не смогли разобраться в том, что нужно изменить, чтобы настроить конфигурацию *vtund.conf* для вашего собственного клиента, обратите внимание на автоматический генератор файла *vtund.conf*, описанный в разделе *«Автоматический генератор файла vtund.conf»* **[Совет № 53]**.

Примечания:

- Название сессии (home в вышеприведенном примере) должно быть одинаковым как со стороны клиента, так и со стороны сервера, иначе вы столкнетесь с двусмысленным сообщением «server is disconnected» (нет связи с сервером).

- То же самое касается поля password в файле *vtund.conf*. Оно должно присутствовать и на сервере, и на клиенте и иметь одно и то же значение, иначе подключение не будет установлено.
- Если возникли проблемы с подключением, убедитесь, что на обеих системах используется одна и та же версия ядра, а сервер запущен и является активным (запустите команду telnet server 5000 на системе клиента, чтобы убедиться в работоспособности сервера).
- Вначале воспользуйтесь прямым методом: затем, удостоверившись в правильности настроек *vtund.conf*, начинайте работать с ssh.
- Если проблемы остаются, проверьте файл */etc/syslog.conf*, чтобы узнать, куда уходят сообщения средств аутентификации auth. Просмотрите, что записывается в файл журнала на обеих системах во время попытки установки подключения.

Дополнительная информация:

- домашняя страница программы vtun: *http://vtun.sourceforge.net/*;
- */usr/src/linux/Documentation/networking/tuntap.txt*;
- оперативное руководство по vtund и vtund.conf (команды man vtund и man vtund.conf, соответственно);
- *«Автоматический генератор файла vtund.conf»* **[Совет № 53]**;
- *«Создание брандмауэра из командной строки любого сервера»* **[Совет № 45]**;
- *«Переназначение портов через ssh»* **[Совет № 71]**.

СОВЕТ №53

Автоматический генератор файла vtund.conf

(Генерирование файла vtund.conf «на лету» для приведения его в соответствие с изменившимися сетевыми условиями)

Вы только что прочли раздел *«Туннелирование: инкапсуляция GRE»* **[Совет № 51]** и знаете, что этот сценарий позволяет создавать рабочий файл *vtund.conf* для клиента автоматически.

Если же вы не читали его, вернитесь назад и прочтите данный параграф, чтобы серьезно разобраться с этим фрагментом кода на языке Perl. Фактически тут требуется воспользоваться угадыванием для изменения таблицы маршрутизации со стороны клиента путем автоматического определения шлюза по умолчанию и создания соответствующего файла *vtund.conf*.

Обратитесь к содержащемуся в сценарии разделу настроек. В первой строке $Config содержатся имена и адреса узлов, номера портов и способы защиты информации: все то, что использовалось нами в разделе *«Туннелирование: инкапсуляция GRE»* **[Совет № 51]**. Вторая строка в данном случае просто служит примером добавления дополнительных настроек.

Чтобы выполнить сценарий, запустите команду vtundconf home или задайте в $TunnelName то имя, которое хотите использовать по умолчанию, или — создайте символические ссылки на сценарий:

```
# ls --s vtundconf home
# ls --s vtundconf tunnel2
```

Сгенерируйте нужный файл *vtund.conf*, обратившись непосредственно к символической ссылке:

```
# vtundconf home > /usr/local/etc/vtund.conf
```

Зачем бороться с подобными трудностями, чтобы написать сценарий для генерирования файла *vtund.conf*? Ведь как только мы добьемся правильных настроек, их больше не придется менять, не так ли?

Да, в большинстве случаев это действительно так. Но, предположим, есть ноутбук под управлением Linux, с помощью которого вы в течение дня подключаетесь к самым разным сетям (скажем, к DSL-соединению дома, сети Ethernet на работе и, может быть, к беспроводной сети в местной кофейне). Запуская сценарий *vtund.conf* в том или ином окружении, можно практически моментально получать рабочие настройки, даже если ваш IP-адрес и шлюз назначаются через протокол DHCP. Это значительно облегчает быстрое подключение к сети с получением рабочего, трассируемого IP-адреса, независимо от топологии этой сети.

Сценарии *vtund* и *vtund.conf* ныне могут быть запущены в операционных системах Linux, FreeBSD, OS X, Solaris и некоторых других.

Листинг: файл vtund.conf

```
#!/usr/bin/perl -w

# упаковщику vtund требуется лучшее название.
#
# (c)2002 Schuyler Erle & Rob Flickenger
#
################ НАСТРОЙКИ

# Если TunnelName не задано, упаковщик будет следить за @ARGV или $0.
#
# В параметре Config задается TunnelName, LocalIP, RemoteIP, TunnelHost,
# TunnelPort, Secret
#
my $TunnelName      = "";
my $Config    = q{
home 208.201.239.33 208.201.239.32 208.201.239.5 5000 sHHH
tunnel2 10.0.1.100 10.0.1.1 192.168.1.4 6001 foobar
};
```

```
################ ОСНОВНАЯ ЧАСТЬ ПРОГРАММЫ

use POSIX 'tmpnam';
use IO::File;
use File::Basename;
use strict;

# Где что искать...
#
$ENV{PATH} = "/bin:/usr/bin:/usr/local/bin:/sbin:/usr/sbin:/usr/local/sbin";
my $IP_Match = '((?:\d{1,3}\.){3}\d{1,3})'; # соответствие xxx.xxx.xxx.xxx
my $Ifconfig = "ifconfig -a";
my $Netstat = "netstat -rn";
my $Vtund = "/bin/echo";
my $Debug = 1;

# Загрузим шаблон из раздела данных.
#
my $template = join( "", <DATA> );

# Откроем временный файл – программный код взят из книги Perl Cookbook,
# 1-е изд., параграф. 7.5.
#
my ( $file, $name ) = ("", "");
$name = tmpnam() until $file = IO::File->new( $name, O_RDWR|O_CREAT|O_EXCL );
END { unlink( $name ) or warn "Can't remove temporary file $name!\n"; }
# Не могу удалить временный файл

# Если TunnelName не задано, используем первый параметр
# из командной строки
# либо, если параметры отсутствуют, базовое имя сценария.
# Т.о., пользователи получают возможность создавать символические
# ссылки на один и тот же сценарий из различных туннелей.
#
$TunnelName ||= shift(@ARGV) || basename($0);
die "Can't determine tunnel config to use!\n" unless $TunnelName;
# Невозможно определить конфигурацию туннеля!

# Выполняем грамматический разбор параметра config.
#
my ($LocalIP, $RemoteIP, $TunnelHost, $TunnelPort, $Secret);
for (split(/\r*\n+/, $Config)) {
my ($conf, @vars) = grep( $_ ne "", split( /\s+/ ));
next if not $conf or $conf =~ /^\s*#/o;
# пропускаем пустые строки и комментарии
if ($conf eq $TunnelName) {
($LocalIP, $RemoteIP, $TunnelHost, $TunnelPort, $Secret) = @vars;
last;
}
}

die "Can't determine configuration for TunnelName '$TunnelName'!\n"
# Невозможно определить конфигурацию для TunnelName
unless $RemoteIP and $TunnelHost and $TunnelPort;
```

```
# Ищем шлюз по умолчанию.
#
my ( $GatewayIP, $ExternalDevice );

for (qx{ $Netstat }) {
# И в Linux, и в BSD шлюз задается в следующей строке,
# а сам интерфейс – в последней строке.
#
if ( /^(?:0.0.0.0|default)\s+(\S+)\s+.*?(\S+)\s*$/o ) {
$GatewayIP = $1;
$ExternalDevice = $2;
last;
}
}

die "Can't determine default gateway!\n" unless $GatewayIP and $ExternalDevice;
# Невозможно определить шлюз по умолчанию!

# Вычисляем LocalIP и LocalNetwork.
#
my ( $LocalNetwork );
my ( $iface, $addr, $up, $network, $mask ) = "";

sub compute_netmask {
 ($addr, $mask) = @_;
# Задаем для $addr маску $mask, в противном случае linux
# /sbin/route предупредит нас о несоответствии сетевого адреса
# маске сети.
#
my @ip = split( /\./, $addr );
my @mask = split( /\./, $mask );
$ip[$_] = ($ip[$_] + 0) & ($mask[$_] + 0) for (0..$#ip);
$addr = join(".", @ip);
return $addr;
}

for (qx{ $Ifconfig }) {
last unless defined $_;

# Если появилось новое устройство, скроем предыдущее
# (если таковое имеется).
if ( /^([^\s:]+)/o ) {
if ( $iface eq $ExternalDevice and $network and $up ) {
$LocalNetwork = $network;
last;
}
$iface = $1;
$up = 0;
}

# Вычислим сетевую маску для текущего интерфейса.
if ( /addr:$IP_Match.*?mask:$IP_Match/io ) {
# ifconfig в стиле Linux.
compute_netmask($1, $2);
$network = "$addr netmask $mask";
```

```
} elsif ( /inet $IP_Match.*?mask 0x([a-f0-9]{8})/io ) {
# ifconfig в стиле BSD.
($addr, $mask) = ($1, $2);
$mask = join(".", map( hex $_, $mask =~ /(..)/gs ));
compute_netmask($addr, $mask);
$network = "$addr/$mask";
}

# Игнорируем интерфейсы устройств обратной петли и неактивных устройств
$iface = "" if /\bLOOPBACK\b/o;
$up++ if /\bUP\b/o;
}

die "Can't determine local IP address!\n" unless $LocalIP and $LocalNetwork;
# Невозможно определить локальный IP-адрес

# Задание переменных, зависимых от ОС.
#
my ( $GW, $NET, $PTP );
if ( $^O eq "linux" ) {
$GW = "gw"; $PTP = "pointopoint"; $NET = "-net";
} else {
$GW = $PTP = $NET = "";
}

# Выполним грамматический разбор шаблона config.
#
$template =~ s/(\$\w+)/$1/gee;

# Запишем временный файл и выполним сценарий vtund.
#
if ($Debug) {
print $template;
} else {
print $file $template;
close $file;
system("$Vtund $name");
}

__DATA__

options {
port $TunnelPort;
ifconfig /sbin/ifconfig;
route /sbin/route;
}

default {
compress no;
speed 0;
}
$TunnelName { # в качестве 'mytunnel' вообще-то должно выступать
# `basename $0` либо нечто похожее
# для автоматического выбора конфигурации
type tun;
```

```
proto tcp;
keepalive yes;

pass $Secret;

up {
ifconfig "%% $LocalIP $PTP $RemoteIP arp";
route "add $TunnelHost $GW $GatewayIP";
route "delete default";
route "add default $GW $RemoteIP";
route "add $NET $LocalNetwork $GW $GatewayIP";
};

down {
ifconfig "%% down";
route "delete default";
route "delete $TunnelHost $GW $GatewayIP";
route "delete $NET $LocalNetwork";
route "add default $GW $GatewayIP";
};
}
```

Мониторинг

Советы № 54-65

Не зная, как система ведет себя в нормальных условиях, весьма сложно иметь представление о ее «тонких» настройках. Осторожно задавая системе вопросы и соответствующим образом интерпретируя ответы, можно избежать блуждания впотьмах и внести эффективные поправки именно там, где они наиболее необходимы.

Здесь на помощь придут файлы журналов. Их тщательный анализ позволит получить массу информации о том, как работает система. Если же создаваемые журналы будут только занимать место на диске, они могут стать источником неразберихи. При использовании надлежащих приемов и средств системные журналы могут стать лаконичным и в то же время достаточно подробным источником всей необходимой информации для настроек.

Но иногда оказывается, что сведения, которые больше всего нужны в данный момент, нигде не запротоколированы, хотя их можно извлечь из ОС Linux путем выборочной проверки системных ресурсов либо данных в самой сети. В этой главе не производится анализ системного мониторинга или пакетов анализа тенденций (таких как Nagios или MRTG), а рассмотрены способы опроса системы о происходящем в *данный момент времени*.

Здесь показано несколько способов выявления потенциальных проблем до момента их возникновения, а также то, как автоматически справляться с катастрофическими сбоями, если они все-таки произошли.

СОВЕТ №54

Управление службой syslog

(Не тратьте время на просмотр огромных файлов журнала — переложите работу на плечи службы syslog)

Устанавливаемая по умолчанию служба syslog не очень хорошо справляется с задачей сортировки различных классов информации между отдельными файлами. Если вы находите сообщения от sendmail,

sudo, bind и других системных служб среди сообщений файла */var/log/*, следует проверить файл настроек */etc/syslog.conf*.

Сортировка сообщений демоном syslog может производиться по категориям и приоритетам. В качестве справочной приведем следующую информацию:

Категории	Приоритеты
auth	debug
auth-priv	info
cron	notice
daemon	warning
kern	err
lpr	crit
mail	alert
news	emerg
syslog	
user	
uucp	
local0-local7	

Учтите, что приложения решают самостоятельно, сообщения каких категорий и какого приоритета должны записываться в журнал, и только самые лучшие приложения позволяют делать этот выбор вам: поэтому не всегда запись будет происходить так, как ожидается. Ниже приводится пример файла настроек */etc/syslog.conf*, с помощью которого организуется запись различной информации в разные файлы журнала:

```
auth.warning /var/log/auth
mail.err /var/log/maillog
kern.* /var/log/kernel
cron.crit /var/log/cron
*.err;mail.none /var/log/syslog
*.info;auth.none;mail.none /var/log/messages

#*.=debug /var/log/debug

local0.info /var/log/cluster
local1.err /var/log/spamerica
```

В соответствующие файлы журнала будет осуществляться запись сообщений с заданным (либо более высоким) приоритетом. Специальный приоритет none означает, что демон syslog вообще не будет сохранять сообщений данной категории. Категории local0-local7 предназначены для обозначения ваших собственных программ в случае возникновения такой необходимости. К примеру, в файл */var/log/spamerica* будут записы-

ваться сообщения категории local1 с приоритетом err (или более высоким), генерируемые программой по отсеву нежелательной почты. Подобные сообщения лучше хранить отдельно от основного журнала почтовой доставки (в */var/log/maillog*).

Закомментированная строка *.=debug нужна для отладки служб, запущенных в роли демонов. Она сообщает syslog о необходимости вести запись сообщений с приоритетом отладки (debug), поступающих от любых служб. Используйте эту функцию только в случае необходимости, если не хотите переполнить все дисковое пространство подобными сообщениями. Другой подход состоит в записи отладочной информации в буфер fifo. В этом случае журналы отладки не будут занимать место на диске и будут удалены, как только процесс перестанет их контролировать. Для записи журнала в fifo необходимо вначале создать его в файловой системе:

```
# mkfifo -m 0664 /var/log/debug
```

Затем исправьте в файле *syslog.conf* следующую строку, добавив символ |:

```
*.=debug |/var/log/debug
```

Теперь отладочная информация будет записываться в буфер fifo, а просмотреть ее в дальнейшем можно при помощи команды, типа less -f /var/log/debug. Весьма удобно настроить ведение журнала для всех системных сообщений, а также, возможно, функцию уведомления по электронной почте при появлении критичных системных сообщений. Попробуйте создать fifo под названием */var/log/monitor* и добавить в конфигурационный файл *syslog.conf* такое правило:

```
*.* |/var/log/monitor
```

Теперь любое сообщение (с любым приоритетом) будет передаваться в fifo */var/log/monitor*, и любой наблюдающий процесс будет мгновенно реагировать соответствующим образом, не занимая при этом пространство на диске.

Отмечаем кого?

Обратите внимание на следующие строки в файле */var/log/messages*:

```
Dec 29 18:33:35 catlin -- MARK --
Dec 29 18:53:35 catlin -- MARK --
Dec 29 19:13:35 catlin -- MARK --
Dec 29 19:33:35 catlin -- MARK --
Dec 29 19:53:35 catlin -- MARK --
Dec 29 20:13:35 catlin -- MARK --
Dec 29 20:33:35 catlin -- MARK --
```

```
Dec 29 20:53:35 catlin -- MARK --
Dec 29 21:53:35 catlin -- MARK --
```

Они генерируются функцией выделения службы syslog путем взаимодействия с системой, так что теоретически можно определять момент прекращения работы этой службы. В большинстве случаев эта функция только переполняет файлы журнала: посему если никаких проблем со службой syslog не наблюдается, то она просто не нужна. Для ее отключения задайте в командной строке syslogd параметр -m-0 (отключив вначале службу syslogd):

```
# killall syslogd; /usr/sbin/syslogd -m -0
```

Удаленное ведение журнала

В современных версиях службы syslogd, по понятным причинам, отключена возможность загрузки журналов с удаленных компьютеров. Без подобных защитных мер для случайного злоумышленника не составит труда переполнить ваш диск фальшивыми файлами журналов, поскольку в этой службе не предусмотрены функции аутентификации узлов сети. В то же время, очень удобно запускать ее централизованно, особенно если приходится работать с целым кластером компьютеров.

Чтобы разрешить загрузку журналов с удаленных систем, запустите демон syslogd с флажком -r:

```
# /usr/sbin/syslogd -m 0 -r
```

На каждом компьютере, журналы с которого должны направляться на ведущий сервер, в файл *syslog.conf* понадобится добавить следующую строку:

```
*.* @master.syslog.host.com
```

Естественно, необходимо задать имя собственного узла либо его IP-адрес после символа @. Желательно защитить порт syslog на сервере службы syslogd. Эта служба прослушивает UDP-порт под номером 514, и можно задать правило фильтрации при помощи команды iptables:

```
# iptables -A INPUT -t filter -p udp --dport 514 -s ! $LOCAL_NET -j DROP
```

где параметр $LOCAL_NET представляет собой сеть либо узел, с которых будут приниматься сообщения службы syslog: например, 192.168.1.0/24 либо имя florian.nocat.net.

Улучшение организации системных журналов облегчит поиск нужной информации. Более детальную информацию по работе с файлами журналов можно найти в разделе *«Анализ журналов, выводимых на экран с цветовым форматированием»* **[Совет № 75]**.

СОВЕТ №55

Наблюдение за выполнением заданий при помощи службы watch

(Используйте службу watch для повторного запуска различных команд и вывода результатов)

Если вы когда-нибудь имели дело с долго выполняющимися процессами, то, безусловно, использовали команды grep или ps для проверки *состояния процесса*:

```
mc@escher:~$ ps ax |grep tar
10303 ? S 0:00 bash -c cd /; tar cf – home
10304 ? S 0:42 tar cf – home
mc@escher:~$ ps ax |grep tar
10303 ? S 0:00 bash -c cd /; tar cf – home
10304 ? S 0:43 tar cf – home
```

Возможно, вы использовали команду ftping для файла, но забыли о команде hash (и, по ряду причин, не использовали ncftp, scp, wget или curl), то есть не имели ни малейшего представления о длительности выполнения. Поэтому, естественно, приходилось входить в систему повторно и проверять содержимое файла при помощи команды ls:

```
mc@escher:~$ ls -l xfree86.tgz
-rw-r--r-- 1 rob users 12741812 Jun 13 2001 xfree86.tgz
mc@escher:~$ ls -l xfree86.tgz
-rw-r--r-- 1 rob users 12744523 Jun 13 2001 xfree86.tgz
```

В ситуациях, когда возникает необходимость повторного запуска одной и той же команды, попробуйте использовать команду watch. Она позволяет периодически запускать нужную команду через заданные промежутки времени (по умолчанию, через каждые 2 секунды). После каждого запуска команда очищает экран, делая выводимую информацию удобной для чтения.

```
mc@escher:~$ watch 'ps ax |grep tar'

Every 2s: ps ax |grep tar|grep -v pts/0 Fri Sep 6 00:22:01 2002

10303 ? S 0:00 bash -c cd /; tar cf – home
10304 ? S 0:42 tar cf – home
```

Если используются каналы, понадобится заключить вызов команды в одинарные кавычки либо другие спецсимволы, которые не должны изменяться перед запуском команды watch. Для задания другого временного интервала воспользуйтесь переключателем -n:

```
mc@escher:~$ watch -n1 ls -l xfree86.tgz

Every 1s: ls -l xfree86.tgz Fri Sep 6 00:31:41 2002

-rw-r--r-- 1 rob users 12756042 Jun 13 2001 xfree86.tgz
```

При каждом проходе различия будут отображаться на экране инверсным цветом. Для этого необходимо воспользоваться переключателем -d (особенно хорошо он подходит для просмотра результатов выполнения команды netstat -a | grep ESTAB с целью наблюдения за установкой и разрывом сетевых подключений). Для завершения работы программы достаточно нажать ^C.

Дополнительная информация:

- оперативное руководство по программе watch (команда man watch).

СОВЕТ №56

Что удерживает порт в открытом состоянии?

(Определите с помощью netstat, какой процесс с каким портом взаимодействует)

Генерирование списка активных портов на сервере Linux производится при помощи утилиты netstat:

```
root@catlin:~# netstat -ln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State
tcp 0 0 0.0.0.0:5280 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:80 0.0.0.0:* LISTEN
tcp 0 0 10.42.3.2:53 0.0.0.0:* LISTEN
tcp 0 0 10.42.4.6:53 0.0.0.0:* LISTEN
tcp 0 0 127.0.0.1:53 0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN
udp 0 0 10.42.3.2:53 0.0.0.0:*
udp 0 0 10.42.4.6:53 0.0.0.0:*
udp 0 0 127.0.0.1:53 0.0.0.0:*
udp 0 0 0.0.0.0:67 0.0.0.0:*
raw 0 0 0.0.0.0:1 0.0.0.0:*7
```

Здесь мы видим обычные службы: веб-сервер, работающий с портом 80, DNS — с портом 53, ssh — с портом 22, dchp — с портом 67, но что за процесс работает с портом 5280?

Выяснить, какие программы на самом деле работают с теми или иными портами, очень просто при помощи последних версий программы netstat. Если вы обладаете правами пользователя root, добавьте переключатель -p для вывода информации по программам:

```
root@catlin:~# netstat -ln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:5280 0.0.0.0:* LISTEN 698/perl
tcp 0 0 0.0.0.0:80 0.0.0.0:* LISTEN 217/httpd
tcp 0 0 10.42.3.2:53 0.0.0.0:* LISTEN 220/named
tcp 0 0 10.42.4.6:53 0.0.0.0:* LISTEN 220/named
tcp 0 0 127.0.0.1:53 0.0.0.0:* LISTEN 220/named
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 200/sshd
udp 0 0 0.0.0.0:32768 0.0.0.0:* 220/named
udp 0 0 10.42.3.2:53 0.0.0.0:* 220/named
udp 0 0 10.42.4.6:53 0.0.0.0:* 220/named
udp 0 0 127.0.0.1:53 0.0.0.0:* 220/named
udp 0 0 0.0.0.0:67 0.0.0.0:* 222/dhcpd
raw 0 0 0.0.0.0:1 0.0.0.0:* 7 222/dhcpd
```

Вот это уже лучше. Идентификатор программы 698 соответствует процессу Perl, связанному с портом 5280. Его можно найти при помощи команды ps:

```
root@catlin:~# ps auwex |grep -w 698
nocat 698 0.0 2.0 5164 3840 ? S Aug 25 0:00 /usr/bin/perl -w ./bin/gateway
PWD=/usr/local/nocat HOSTNAME=catlin.r
```

Команда ps auwex позволяет получать сведения обо всех (a) неинтерактивных (x) процессах, включая информацию о пользователях (u) в расширенном формате (w) с дополнительными битами окружения (e).

Затем применим команду grep к границам слов (w) для PID.

Еще лучше: теперь нам известно, что пользователь nocat запустил в каталоге */usr/local/nocat/* команду */bin/gateway*, сценарий на языке Perl, прослушивающий порт 5280. Без помощи переключателя -p команды netstat вычислить, какой порт с каким процессом связан, несколько сложнее.

Если вы не являетесь пользователем root, то не сможете узнать, какие программы на каких портах запущены, а получите следующее сообщение об ошибке:

```
(No info could be read for "-p": getuid = 1000 but you should be root.)
```

В таком случае сначала заполучите права пользователя root, а потом уже можете запускать вышеупомянутую команду! Можно также обратиться к программе sudo, описанной в разделе *«Заставим программу sudo работать не покладая рук»* **[Совет № 12]**, чтобы произвести запуск этой команды от имени пользователя root.

СОВЕТ №57

Проверка открытых файлов и сокетов с помощью lsof

(Проверьте с помощью lsof, какие файлы, директории и сокеты остаются открытыми)

Вы когда-нибудь пробовали демонтировать файловую систему только для того, чтобы впоследствии обнаружить, что некоторые процессы ее все еще используют?

```
root@mouse:~# umount /mnt
umount: /mnt: device is busy
```

Для определения процессов, использующих файловую систему */mnt*, попробуйте воспользоваться утилитой lsof:

```
root@mouse:~# lsof /mnt
COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
bash 30951 rob cwd DIR 7,0 1024 2 /mnt
```

Как видим, для пользователя rob в данный момент текущей является директория */mnt*, т.к. оболочка bash установила для нее атрибуты cwd. Утилита lsof выводит список всех открытых файлов, директорий, библиотек, сокетов и устройств, связанных с данным процессом. В приведенном выше примере мы задали точку монтирования файловой системы, заставив команду lsof вывести информацию по всем связанным процессам. Чтобы добиться обратного эффекта — вывести список файлов, связанных с идентификатором процесса (PID) — воспользуйтесь переключателем -p:

```
root@mouse:~# lsof -p 30563
COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
inetd 30563 root cwd DIR 3,3 408 2 /
inetd 30563 root rtd DIR 3,3 408 2 /
inetd 30563 root txt REG 3,3 21432 39140 /usr/sbin/inetd
inetd 30563 root mem REG 3,3 432647 11715 /lib/ld-2.2.3.so
inetd 30563 root mem REG 3,3 4783716 11720 /lib/libc-2.2.3.so
inetd 30563 root mem REG 3,3 19148 11708 /lib/libnss_db-2.2.so
inetd 30563 root mem REG 3,3 238649 11728 /lib/libnss_files-2.2.3.so
inetd 30563 root mem REG 3,3 483324 11710 /lib/libdb-3.1.so
inetd 30563 root 0u CHR 1,3 647 /dev/null
inetd 30563 root 1u CHR 1,3 647 /dev/null
inetd 30563 root 2u CHR 1,3 647 /dev/null
inetd 30563 root 4u IPv4 847222 TCP *:telnet (LISTEN)
inetd 30563 root 5u IPv4 560439 TCP *:cvspserver (LISTEN)
```

Для указания имени процесса используйте переключатель -c:

```
root@mouse:~# lsof -c syslogd
COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
syslogd 25627 root cwd DIR 3,3 408 2 /
syslogd 25627 root rtd DIR 3,3 408 2 /
syslogd 25627 root txt REG 3,3 27060 5538 /usr/sbin/syslogd
```

```
syslogd 25627 root mem REG 3,3 432647 11715 /lib/ld-2.2.3.so
syslogd 25627 root mem REG 3,3 4783716 11720 /lib/libc-2.2.3.so
syslogd 25627 root mem REG 3,3 238649 11728 /lib/libnss_files-2.2.3.so
syslogd 25627 root mem REG 3,3 75894 11719 /lib/libnss_dns-2.2.3.so
syslogd 25627 root mem REG 3,3 225681 11724 /lib/libresolv-2.2.3.so
syslogd 25627 root mem REG 3,3 19148 11708 /lib/libnss_db-2.2.so
syslogd 25627 root mem REG 3,3 483324 11710 /lib/libdb-3.1.so
syslogd 25627 root 0u unix 0xdcc2d5b0 775254 /dev/log
syslogd 25627 root 1w REG 3,3 135744 5652 /var/log/debug
syslogd 25627 root 2w REG 3,3 107459 5651 /var/log/syslog
syslogd 25627 root 4w REG 3,3 107054 58317 /var/log/maillog
syslogd 25627 root 5w REG 3,3 4735 16 TCP /var/log/authlog
```

Кроме того, можете задавать из командной строки специальные устройства: например, увидеть, что собирается делать пользователь терминала pts/0:

```
root@mouse:~# lsof /dev/pts/0
COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
bash 29816 rob 0u CHR 136,0 2 /dev/pts/0
bash 29816 rob 1u CHR 136,0 2 /dev/pts/0
bash 29816 rob 2u CHR 136,0 2 /dev/pts/0
bash 29816 rob 255u CHR 136,0 2 /dev/pts/0
lspf 30882 root 0u CHR 136,0 2 /dev/pts/0
lspf 30882 root 1u CHR 136,0 2 /dev/pts/0
lspf 30882 root 2u CHR 136,0 2 /dev/pts/0
```

Если требуется задать несколько переключателей, между ними по умолчанию устанавливаются отношения ИЛИ. Чтобы включить использование всех переключателей (то есть, установить для них логическое объединение И), вставьте флажок -a перед каждым переключателем, который вы хотите использовать по команде И. Например, чтобы увидеть список всех открытых файлов, связанных с процессами vi, наберите следующую команду:

```
root@mouse:~# lsof -u rob -ac vi
COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
vi 31059 rob cwd DIR 3,3 2894 39681 /home/rob
vi 31059 rob rtd DIR 3,3 408 2 /
vi 31059 rob txt REG 3,3 554504 9799 /usr/bin/vim
vi 31059 rob mem REG 3,3 432647 11715 /lib/ld-2.2.3.so
vi 31059 rob mem REG 3,3 282178 2825 /lib/libncurses.so.5.2
vi 31059 rob mem REG 3,3 76023 2831 /usr/lib/libgpm.so.1.18.0
vi 31059 rob mem REG 3,3 4783716 11720 /lib/libc-2.2.3.so
vi 31059 rob mem REG 3,3 249120 11721 /lib/libnss_compat-2.2.3.so
vi 31059 rob mem REG 3,3 357644 11725 /lib/libnsl-2.2.3.so
vi 31059 rob 0u CHR 136,1 3 /dev/pts/1
vi 31059 rob 1u CHR 136,1 3 /dev/pts/1
vi 31059 rob 2u CHR 136,1 3 /dev/pts/1
vi 31059 rob 3u REG 3,3 4096 15 /home/rob/.sushi.c.swp
```

Если требуется проверить, какие сокеты открыты и какие процессы связаны с ними, как это делается с помощью команды netstat -p, воспользуйтесь переключателем -i:

```
root@mouse:~# lsof -i
COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
sshd 69 root 3u IPv4 61 TCP *:ssh (LISTEN)
mysqld 126 mysql 3u IPv4 144 TCP *:3306 (LISTEN)
mysqld 128 mysql 3u IPv4 144 TCP *:3306 (LISTEN)
mysqld 129 mysql 3u IPv4 144 TCP *:3306 (LISTEN)
httpd 24905 root 22u IPv4 852520 TCP *:443 (LISTEN)
httpd 24905 root 23u IPv4 852521 TCP *:www (LISTEN)
httpd 28383 www 4u IPv4 917713 TCP nocat.net:www->65.192.187.158:8648
(ESTABLISHED)
httpd 28389 www 4u IPv4 917714 TCP nocat.net:www->65.192.187.158:9832
(ESTABLISHED)
httpd 28389 www 22u IPv4 852520 TCP *:443 (LISTEN)
httpd 28389 www 23u IPv4 852521 TCP *:www (LISTEN)
exim 29879 exim 0u IPv4 557513 TCP *:smtp (LISTEN)
inetd 30563 root 4u IPv4 847222 TCP *:telnet (LISTEN)
inetd 30563 root 5u IPv4 560439 TCP *:cvspserver (LISTEN)
sshd 30973 root 4u IPv4 901571 TCP nocat.net:ssh->some.where.net:52543
(ESTABLISHED)
sshd 30991 root 4u IPv4 901577 TCP nocat.net:ssh->some.where.else:52544
(ESTABLISHED)
```

Учтите, что для использования большинства функций команды lsof, включая получение информации об открытых сокетах, необходимо обладать правами пользователя root. В целом же lsof представляет собой сложную, но чрезвычайно гибкую утилиту, позволяющую получать настолько подробную информацию о файлах, используемых различными процессами системы, насколько это необходимо.

Дополнительная информация:

- последнюю версию утилиты lsof можно загрузить с сайта *ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/*.

СОВЕТ №58

Мониторинг системных ресурсов с помощью утилиты top

(Воспользуйтесь утилитой top для определения состояния системы)

Команда top выдает сведения о текущей загрузке системы, использовании памяти и процессора. Она входит в состав пакета утилит procps. Простейший способ ее запуска — набрать имя утилиты в командной строке:

```
$ top
```

В результате весь экран окажется заполненным информацией, обновляемой каждую пару секунд.

```
3:54pm up 1 day, 16 min, 2 users, load average: 0.00, 0.00, 0.00
38 processes: 37 sleeping, 1 running, 0 zombie, 0 stopped
CPU states: 0.0% user, 0.7% system, 0.0% nice, 99.2% idle
Men 189984k av, 155858 used, 34116K free, 0K shrd, 42444K buff
Swap: 257032K av, 0K used, 257032K free 60028K cached

COMMAND PID USER FD TYPE DEVICE SIZE NODE NAME
6195 rob 14 0 1004 1004 800 R 0.5 0.5 0:00 top
1 root 8 0 212 212 180 S 0.0 0.1 0:13 init
2 root 9 0 0 0 0 SW 0.0 0.0 0:00 reventd
3 root 9 0 0 0 0 SW 0.0 0.0 0:00 kswapd
4 root 9 0 0 0 0 SW 0.0 0.0 0:00 kreclaimd
5 root 9 0 0 0 0 SW 0.0 0.0 0:00 bdflush
6 root 9 0 0 0 0 SW 0.0 0.0 0:00 kupdated
8 root -1 -20 0 0 0 SW< 0.0 0.0 0:00 mdrecoveryd
176 root 9 0 788 788 680 S 0.0 0.4 0:00 syslogd
179 root 9 0 1228 1228 444 S 0.0 0.6 0:00 klogd
182 root 8 0 1228 1228 1104 S 0.0 0.6 0:06 sshd
184 root 8 0 616 616 520 S 0.0 0.3 0:00 crond
186 daemon 9 0 652 652 560 S 0.0 0.3 0:00 atd
197 root 9 0 2544 2544 2396 S 0.0 1.3 0:00 httpd
200 root 9 0 3740 3740 1956 S 0.0 1.9 0:00 named
202 root 9 0 1004 1004 828 S 0.0 0.5 0:00 dhcpd
203 root 9 0 504 504 444 S 0.0 0.2 0:00 agetty
```

Нажмите ? во время работы утилиты top, чтобы увидеть список доступных параметров. Запомните полезные клавиши, управляющие отображением результатов: M – выполняет сортировку результатов по размеру занимаемой ими резидентной памяти, P – сортирует процессы по степени загрузки процессора, S – сортирует по суммарному времени работы родительского и всех его дочерних процессов в секундах использования ЦПУ, i – отключает вывод сведений о неактивных процессах.

Если утилита top запускается с правами пользователя root, в ваше распоряжение не поступят еще несколько интерактивных команд и команд сортировки, которые могут оказаться полезными. Флажок u, например, позволяет отфильтровывать все процессы, за исключением тех, запуск которых был инициирован данным пользователем. Флажок k позволяет интерактивно уничтожать заданный процесс по его PID с любым выбранным типом уведомления. Эта функция подходит для выявления вышедших из-под контроля процессов, позволяя уничтожать их прямо из команды top (можете просто скопировать и вставить заинтересовавший вас идентификатор процесса (PID), чтобы избежать ужасных последствий опечатки).

Очень удобно оставлять утилиту top работающей на окнах терминала загруженных систем, к которым вы только подключены, но не рабо-

таете с ними. О том, как постоянно видеть среднюю загрузку системы в титульной строке окна приветствия (типа мини-окна утилиты top, оставаясь внутри оболочки), рассказывается в разделе *«Постоянное отображение средней загрузки в строке заголовка»* **[Совет № 59]**.

Дополнительная информация:

- *«Оперирование символьными именами процессов в procps»* **[Совет № 17]**;
- *«Постоянное отображение средней загрузки системы в строке заголовка»* **[Совет № 59]**;
- домашняя страница утилиты procps: *ftp://people.redhat.com/johnsonm/procps/*.

СОВЕТ №59 Постоянное отображение средней загрузки системы в строке заголовка

(Заставьте работать строку заголовка)

Если вы хотя бы некоторое время занимались управлением сервером Linux, то, вероятно, завели близкое знакомство с утилитой top; в противном случае немедленно прекратите чтение и наберите top в командной строке ближайшего терминала, а затем, когда вам наскучит, нажмите ?. Если же вы отвечали за управление сразу несколькими Linux-серверами, то вам должно быть известно, что это эквивалентно запуску нескольких копий утилиты top в нескольких окнах, каждое из которых будет бороться за захват рабочего стола.

Применительно к компьютерным вычислениям, потерянными называют ресурсы, которые могли быть потрачены более продуктивно. Почему бы не запустить процесс, который выводит на терминал информацию об усредненной загрузке системы в режиме реального времени, независимо от работы других программ системы?

Наберите следующий сценарий и сохраните его под именем *tl* в папке *~/bin*:

Листинг: tl

```
#!/usr/bin/perl -w

use strict;
$|++;

my $host=`/bin/hostname`;
chomp $host;

while(1) {
```

```
open(LOAD,"/proc/loadavg") || die "Couldn't open /proc/loadavg: $!\n";
# Невозможно открыть /proc/loadavg

my @load=split(/ /,<LOAD>);
close(LOAD);

print "\033]0;";
print "$host: $load[0] $load[1] $load[2] at ", scalar(localtime);
print "\007";

sleep 2;
}
```

Если потребуется вывести в строке заголовка имя, среднюю загрузку системы либо текущее время, запустите утилиту tl&. Она продолжит свою работу в фоновом режиме, даже если выполняется такая интерактивная программа как vim. Если уже настроено отображение в строке заголовка текущей директории, никаких проблем возникнуть не должно. В случае выполнения команды cd на экран мгновенно будет выведено имя текущей директории, после чего его заменит информация о времени и средней загрузке системы. Хотите проверить, какая директория является текущей, еще раз? Нажмите ENTER в пустой командной строке, и сведения о текущей рабочей папке будут выведены повторно.

Теперь, вместо того чтобы заставлять весь рабочий стол компьютерными терминалами, можете выводить только строки заголовков, имея возможность «беглым взглядом» оценить напряженную работу каждой системы. Если понадобится самому поработать с компьютером, просто отправьте все окна на задний план и приступайте к решению текущих задач. Причем все это возможно без установки на локальной машине какого-либо дополнительного программного обеспечения.

Закончив работу с утилитой tl, не забудьте запустить команду killall tl перед выходом из системы. Либо, если вам лень работать с терминалом, наберите в командной строке следующее:

```
$ echo 'killall tl > /dev/null 2>&1' >> ~/.bash_logout
```

Результатом выполнения команды станет завершение процесса tl и выход из оболочки, для чего вам не придется даже пальцем пошевелить.

СОВЕТ №60

Сетевой мониторинг при помощи утилиты ngrep

(Используйте утилиту grep в своем сетевом интерфейсе для анализа работы пользователей)

Утилита ngrep представляет собой весьма интересное средство перехвата, напоминающее по своей функциональности программы tcpdump и snoop. Она позволяет максимально облегчить выбор пакетов, которые

необходимо выводить на экран, благодаря использованию формата, совместимого с grep (со всеми регулярными выражениями и группой переключателей GNU). Утилита также позволяет конвертировать пакеты в код ASCII или шестнадцатеричный код перед выводом на экран.

Например, чтобы увидеть содержимое http-запросов GET, проходящих через ваш маршрутизатор, воспользуйтесь следующей командой:

```
# ngrep -q GET
```

Если нужна информация только по определенному узлу сети, протоколу либо порту (или же какому-то другому критерию), задайте фильтр bpf, а также шаблон соответствия данных. Синтаксис команды похож на синтаксис утилиты tcpdump:

```
# ngrep -qi rob@nocat.net port 25

T 10.42.4.7:65174 -> 209.204.146.26:25 [AP]
RCPT TO:<rob@nocat.net>..

T 209.204.146.26:25 -> 10.42.4.7:65174 [AP]
250 2.1.5 <rob@nocat.net>... Recipient ok..

T 10.42.4.7:65174 -> 209.204.146.26:25 [AP]
Date: Sun, 8 Sep 2002 23:55:18 -0700..Mime-Version: 1.0 (Apple Message framework
v543)..Content-Type: text/plain; charset=US-ASCII; format=flowed..Subject:
Greetings.....From: John Doe <johnd@somewhere.else.com>..To:
rob@nocat.net..Content-Transfer-Encoding: 7bit..Message-Id: <19DB8C16-C3C1-11D6-
B239-0003936D6AE0@somewhere.else.com>..X-Mailer: Apple Mail v2>....What does
that pgp command you mentioned do again?....Thanks,....--A Friend....
```

Поскольку утилита ngrep выводит данные в STDOUT, можно выполнять последующую обработку результатов, создавая удобный фильтр по выводу информации. Если хотите вести обработку выходных данных самостоятельно, включите переключатель -l для буферизации строк данных. К примеру, если вас интересует, кто из пользователей сети пытается добиться онлайнового доступа, воспользуйтесь приводимым ниже фрагментом кода на языке Perl:

Листинг: go-ogle

```
#!/usr/bin/perl
use Socket;
$|++;

open(NG, "ngrep -d en1 -lqi '(GET|POST).*/(search|find)' |");
print "Go ogle online.\n";
my ($go,$i) = 0;
my %host = ( );
```

```
while(<NG>) {

if(/^T (\d+\.\d+.\d+\.\d+):\d+ -> (\d+\.\d+\.\d+\.\d+):80/) {
$i = inet_aton($1);
$host{$1} ||= gethostbyaddr($i, AF_INET) || $1;
$i = inet_aton($2);
$host{$2} ||= gethostbyaddr($i, AF_INET) || $2;
print "$host{$1} -> $host{$2} : ";
$go = 1;
next;
}
if(/(q|p|query|for)=(.*)?(&|HTTP)/) {
next unless $go;
my $q = $2;
$q =~ s/(\+|&.*)/ /g;
$q =~ s/%(\w+)/chr(hex($1))/ge;
print "$q\n";
$go = 0;
}
if(/^$/) {
next unless $go;
$go = 0;
print "\n";
}
}
```

Этот сценарий назван go-ogle. В нем осуществляется запуск утилиты ngrep, выполняющей поиск запросов GET или POST, которые включают поиск информации по заданному адресу. Результаты будут выглядеть приблизительно следующим образом:

```
Go ogle online.
caligula.nocat.net -> www.google.com : o'reilly mac os x conference
caligula.nocat.net -> s1.search.vip.scd.yahoo.com : junk mail $$$
tiberius.nocat.net -> altavista.com : babel fish
caligula.nocat.net -> caligula.nocat.net -> 166-140.amazon.com : Brazil
livia.nocat.net -> 66.161.12.119 : lart
```

При этом нельзя избежать кодирования строк в запросах (обратите внимание на символ ' в запросе, отправленном на сервер google, и $$$ – в запросе к yahoo). Сценарий также позволяет конвертировать IP-адреса в имена узлов, т.к. в ngrep, судя по всему, подобная функция отсутствует, за счет чего, вероятно, данная утилита позволяет оптимизировать скорость перехвата. Два последних результата представляют особый интерес: запрос Brazil был обслужен на сайте *http://www.imdb.com/*, а последний запрос выполнен на сайте *http://www.dictionary.com*. Очевидно, что теперь IMDB находится в партнерских отношениях с компанией Amazon, а на поисковом сервере сайта Dictionary.com отсутствуют записи типа PTR (pointer – указатель). Просто удивительно, как много всего интересного о мире можно узнать, наблюдая за отправляемыми чужими сетевыми пакетами!

Учтите, что для запуска утилиты ngrep необходимо обладать привилегиями пользователя root, а для получения лучших результатов ее нужно запускать на маршрутизаторе, расположенном на границе вашей сети.

Дополнительная информация:

- оперативное руководство по утилите *ngrep* (команда man ngrep);
- сайт *http://www.packetfactory.net/Projects/ngrep/*.

СОВЕТ №61

Сканирование компьютеров с помощью утилиты nmap

(Учитесь определять начало работы серверов и служб в онлайновом режиме)

Утилита nmap очень полезна для идентификации компьютеров и служб в сети. Она позволяет выполнять различные виды сканирования сети, начиная со стандартного сканирования TCP и UDP и заканчивая более экзотичными методами сканирования, как например, скрытое сканирование TCP SYN, Xmas Tree, NULL-сканирование и еще ряд других опций.

Еще интереснее выглядит программный код для определения типов операционных систем, выполняющий анализ пакетов, возвращенных с целевого компьютера, и сравнивающий результаты с базой данных известных операционных систем. Такой программный код позволяет определять тип удаленной ОС даже без подключения к какой-либо из служб, возвращая при этом приблизительное время работы сканируемой системы.

Для выполнения простого сканирования портов ICMP и определения типа операционной системы воспользуйтесь переключателем -0:

```
rob@catlin:~# nmap -0 caligula

Starting nmap V. 3.00 (www.insecure.org/nmap/ )
Interesting ports on caligula.rob.nocat (10.42.4.7):
(The 1600 ports scanned but not shown below are in state: closed)
/* 1600 портов подверглись сканированию, но не приводятся ниже, поскольку
они находятся в закрытом состоянии
Port State Service
22/tcp open ssh
Remote operating system guess: Mac OS X 10.1 – 10.1.4
/* Предположительная версия операционной системы: Mac OS X 10.1 – 10.1.4
Uptime 5.760 days (since Tue Sep 3 19:14:36 2002)

Nmap run completed -- 1 IP address (1 host up) scanned in 31 seconds
/* Сканирование завершено: проверен 1 IP-адрес (обнаружен 1 активный узел)
```

Если есть свободное время и желание изучить с помощью nmap всю локальную сеть, задайте в командной строке адрес сети или подсети.

Результатом станет выполнение сканирования TCP SYN и определение типов операционных систем для первых 64 адресов в сети 10.42.4.6:

```
root@catlin:~# nmap -0sS 10.42.4.0/26
```

Поскольку утилита nmap осуществляет вывод информации в STOUT, можете сохранить результаты текущего сканирования и сравнить их с результатами предыдущего, чтобы упростить себе составление отчета о различиях. Дабы избежать получения ошибочных результатов, запустим в командной строке сканирование Xmas Tree и grep (как во время выполнения):

```
root@catlin:~# nmap -sX 10.42.4.0/26 | egrep -v '^(Nmap|Starting)' \
> namp.output
```

Затем выполним ту же самую команду позднее (скажем, через несколько дней, в случайный момент времени):

```
root@catlin:~# nmap -sX localhost | egrep -v '^(Nmap|Starting)' \
> namp.output2
```

и применим в контексте команду diff для определения различий:

```
root@catlin:~# diff -c nmap.output*
*** nmap.output Mon Sep 9 14:45:06 2002
—- nmap.output2 Mon Sep 9 14:45:21 2002
***************
**** 1,7 ****

Interesting ports on catlin.rob.nocat (10.42.4.6):
! (The 1598 ports scanned but not shown below are in state: closed)
/* 1598 портов подверглись сканированию, но не приводятся ниже, поскольку они
находятся в закрытом состоянии
Port State Service
22/tcp open ssh
53/tcp open domain
80/tcp open http
—- 1,8 ——

Interesting ports on catlin.rob.nocat (10.42.4.6):
! (The 1597 ports scanned but not shown below are in state: closed)
/* 1597 портов подверглись сканированию, но не приводятся ниже, поскольку они
находятся в закрытом состоянии
Port State Service
+21/tcp open ftp
22/tcp open ssh
53/tcp open domain
80/tcp open http
root@catlin:~#
```

Все выглядит так, будто пользователь catlin с некоторого момента вдруг начал работать с сервером ftp. Данная технология позволяет об-

наруживать новые активные, а также неактивные узлы сети и службы при каждом своем запуске. Имея архив результатов выполнения команды nmap, сохраненный в файлах с именами, отражающими дату и время создания, либо даже записанный в базу данных, вы получите в свое распоряжение журнал состояния всех компьютеров своей сети. Решение и реализацию задачи по превращению утилиты в сценарий оболочки и ее запуску из планировщика Unix мы возлагаем на наших читателей, т.к. это будет небезынтересное и определенно стоящее упражнение.

Дополнительная информация:

- домашняя страница утилиты nmap: *http://www.insecure.org/nmap*.

СОВЕТ №62

Анализ частоты использования диска

(Определите, какие части файловой системы чаще всего подвергаются изменениям)

Как быстро определить, какие файлы и папки чаще всего модифицируются, а какие остаются без изменений месяцами? Для этого достаточно написать специальный сценарий на языке Perl.

Ниже приводится сценарий на языке Perl, выполняющий анализ использования файловой системы. Программа оценивает файловую систему по двум параметрам: дате последнего изменения и дате последнего обращения. Примерный результат выполнения этого сценария выглядит так:

```
% diskage /usr/local

Disk aging analysis for /usr/local:

last num last num
Age (days) modified files accessed files
0 – 30 260403 Kb 817 140303 Kb 6968
31 – 60 11789 Kb 226 23140 Kb 199
61 – 90 40168 Kb 1126 1087585 Kb 31625
91 – 180 118927 Kb 995 0 Kb 0
181 – 365 85005 Kb 1889 0 Kb 0
366 – 9999 734735 Kb 33739 0 Kb 0
——————- ——- ——————- ——-
Total 1251029 Kb 38792 1251029 Kb 38792
```

Вы можете запустить сценарий с опцией -v для вывода списка файлов, которые были изменены в последнее время и к которым обращение производилось в последнюю очередь.

Листинг: diskage

```
#!/usr/local/bin/perl
#
# Генератор отчетов по частоте использования частей файловой системы
# Автор Seann Herdejurgen
#
# Май 1998

use File::Find;

# Инициализация переменных
@levels=(30,60,90,180,365,9999);

# Проверка флажка verbose
if ($ARGV[0] eq "-v") {
$verbose++;
shift(@ARGV);
}

$ARGV[0]=$ENV{'PWD'} if ($ARGV[0] eq "");

foreach $dir (@ARGV) {
foreach $level (@levels) {
$modified{$level}=0;
$accessed{$level}=0;
$mfiles{$level}=0;
$afiles{$level}=0;
}
print("\nDisk aging analysis for $dir:\n\n");
# Анализ давности использования папки
print (" mod acc size file\n") if ($verbose);

# Анализ нужной части файловой системы
find(\&wanted,$dir);

print(" last num last num\n");
print(" Age (days) modified files accessed files\n");
$msize=$asize=$mtotal=$atotal=$lastlevel=0;

foreach $level (@levels) {
printf("%4d - %4d %8d Kb %5d %8d Kb
%5d\n",$lastlevel,$level,$modified{$level}/1024,$mfiles{$level},$accessed{$level}/
1024,$afiles{$level});
$msize+=$modified{$level}/1024;
$asize+=$accessed{$level}/1024;
$mtotal+=$mfiles{$level};
$atotal+=$afiles{$level};
$lastlevel=$level+1;
}
printf(" ---------- ---- ---------- ----\n");
printf(" Total %8d Kb %5d %8d Kb %5d\n",$msize,$mtotal,$asize,$atotal);
}
```

```
exit;

sub wanted {
((($dev,$ino,$mode,$nlink,$uid,$gid,$rdev,$size) = lstat($_));
$mod=int(-M _);
$acc=int(-A _);
foreach $level (@levels) {
if ($mod<=$level) { $modified{$level}+=$size; $mfiles{$level}++; last; }
}
foreach $level (@levels) {
if ($acc<=$level) { $accessed{$level}+=$size; $afiles{$level}++; last; }
}
printf("%4d %4d %6d %s\n",$mod,$acc,$size,$_) if ($verbose);
}
```

СОВЕТ №63

Контроль над IP-адресами

(Контролируйте IP-адреса при помощи ping, bash и других сетевых утилит)

Перенаправление трафика на некоторую систему достаточно легко организовать при помощи циклического DNS, описанного в разделе *«Распределение серверной загрузки при помощи циклического DNS»* **[Совет № 79]**. Но что произойдет, если один из серверов, выступающий в роли получателя этого трафика, вдруг станет недоступен? Далее приводятся одна из схем мониторинга состояния серверов и способ замены одного сервера на другой в случае сбоя.

Во-первых, необходимо четко провести различия между «реальным» IP-адресом сервера и IP-адресом (или адресами), которые на самом деле публикуют содержимое. Для нашего примера выберем два сервера: Pinky и Brain. Сервер Pinky использует адрес 208.201.239.12 в качестве «реального» IP-адреса в eth0, при этом он представлен IP-псевдонимом 208.201.239.36 на eth0:0. Сервер Brain, в свою очередь, использует IP-адрес 208.201.239.13 на eth0 и 208.201.239.37 на eth0:0. Если вы до этого никогда не пользовались IP-псевдонимами, самое время научиться это делать:

```
# ifconfig eth0:0 1.2.3.4
```

Итак, с eth0 оказывается связанным другой IP-адрес (1.2.3.4), вызываемый под именем eth0:0. Раньше, чтобы достичь подобного результата, приходилось добавлять функцию IP-псевдонимов при компиляции ядра; теперь же эта функция, видимо, включена в ядро по умолчанию. Следует запомнить один важный момент, касающийся псевдонимов IP: если интерфейс, к которому привязывается данный псевдоним, будет отключен, перестанут работать все связанные с ним псевдонимы. В качестве псевдонима можно задавать любую строку, состоящую из сочетания букв и цифр, хотя некоторые версии ifconfig отображают только первые 4-5 символов псевдонима при выводе информации об интерфейсах.

Настроив интерфейсы eth0:0 для серверов Pinky и Brain, свяжем с IP-псевдонимами этих серверов некоторую службу, например Apache, и настроим циклический DNS так, чтобы иметь возможность обращаться к любому серверу исключительно при помощи имени узла (см. раздел «Распределение серверной загрузки при помощи циклического DNS» [Совет № 79]). Предположим, что мы настроили дополнительную веб-службу для сайта *www.oreillynet.com*, которая позволяет обращаться к нему как по адресу 208.201.239.36, так и по адресу 208.201.239.37.

Теперь, когда на каждый сервер направляется приблизительно половина трафика, необходимо добиться, чтобы серверы Pinky и Brain некоторым образом следили за состоянием друг друга. Это можно реализовать, применяя команду ping к их реальным IP-адресам и сравнивая результаты. Сохраните следующий программный код в виде сценария, а затем установите его на сервере Pinky:

Листинг: takeover

```
#!/bin/bash
OTHER="brain"
PUBLIC="208.201.239.37"

PAUSE=3

PATH=/bin:/usr/bin:/sbin:/usr/sbin:/usr/local/sbin
MISSED=0

while true; do
if ! ping -c 1 -w 1 $OTHER > /dev/null; then
((MISSED++))
else
if [ $MISSED -gt 2 ]; then
ifconfig eth0:$OTHER down
fi
MISSED=0
fi;

if [ $MISSED -eq 2 ]; then
ifconfig eth0:$OTHER $PUBLIC
#
# ...обсуждение сценария смотрите ниже...
#
fi
sleep $PAUSE;
done
```

Разумеется, понадобится изменить значение параметра OTHER на Pinky, а значение PUBLIC – на 208.201.239.36 в копии сценария, предназначенной для сервера Brain.

Теперь допустим, что сервер Brain неожиданно перестал отвечать на запросы по адресу 208.201.239.17 (скажем, из-за того, что кто-то из технического обслуживающего персонала не туда воткнул сетевой шнур на системном блоке). После трех откликов ping о недоступности узла за дело возьмется сервер Pinky, связав интерфейс eth0: с открытым IP-адресом 208.201.239.37, который до этого обслуживался сервером Brain. В дальнейшем сервер Pinky продолжит наблюдение за реальным IP-адресом сервера Brain, и когда тот возобновит работу, передаст управление ему. Строка ping -c 1 -w 1 означает «отправить один пакет ping и классифицировать ситуацию как превышение периода ожидания в случае отсутствия ответа в течение секунды, независимо от причин». Команда ping возвратит значение, отличное от нуля, если пакет не вернется по истечении одной секунды.

Однако такой прием не позволяет решить задачу в общем. Хотя теперь сервер Pinky будет отвечать на все запросы, направляемые серверу Brain, любой компьютер, входящий в состав той же сети, что и два вышеупомянутых сервера (особенно это касается маршрутизатора, передающего данные в направлении, противоположном тому, в котором передает данные ваш провайдер услуг интернет), будет кэшировать для 208.201.239.37 неверный MAC-адрес. А из-за неправильного MAC-адреса на сервер Pinky не будет поступать сетевой трафик, поскольку данный сервер настроен отвечать только на те пакеты, в которых указан его собственный MAC-адрес. Как же нам сообщить всем компьютерам в сети 208.201.239.0, что MAC-адрес компьютера 208.201.239.37 изменился?

Один из возможных способов заключается в использовании утилиты send_arp от проекта High Availability Linux (широкодоступная система Linux). Эта небольшая, но чрезвычайно удобная утилита позволяет составлять пакеты протокола разрешения адресов (ARP) в соответствии с заданной вами спецификацией и отправлять их на выбранный MAC-адрес в локальной сети. Если в качестве получателя задан весь диапазон адресов (т.е. ff:ff:ff:ff:ff:ff), такой пакет будет являться широковещательным. Хотя большинство маршрутизаторов не обновляют свои таблицы ARP при виде незапрашиваемых широковещательных пакетов ARP, наличие таких пакетов сигнализирует маршрутизатору о необходимости повторной отправки запросов ARP, на которые сервер Pinky обязательно должен ответить. Преимущество использования широковещательных пакетов в том, что сигнал достигает всех пакетов подсети одновременно, в результате чего отпадает необходимость отслеживать MAC-адреса компьютеров во время обновления.

Команда send_arp имеет такой синтаксис: send_arp [IP-адрес источника] [MAC-адрес источника] [IP-адрес получателя] [MAC-адрес получателя]. К при-

меру, при обнаружении сбоев на сервере Brain приведенный выше сценарий мониторинга должен будет запустить следующую команду:

```
send_arp 208.201.239.37 00:11:22:aa:bb:cc 208.201.239.37 ffffffffffff
```

где 00:11:22:aa:bb:cc — аппаратный MAC-адрес интерфейса eth0, принадлежащего серверу Pinky. Сценарий возобновит наблюдение после того, как реальный IP-адрес сервера Brain (208.201.239.17) станет доступен. Когда это произойдет, мы сможем повторно включить интерфейс eth0:brain и заставить сервер Brain самостоятельно заботиться об обновлении кэша ARP, т.к. он должен быть настроен на выполнение этого действия во время загрузки.

В эту технологию можно внести еще массу усовершенствований, кроме одного момента, связанного с тем, что работоспособность сервера 208.201.239.17 не гарантирует доступность адреса 208.201.239.37. К тому же выполнение простого сканирования ICMP при помощи команды ping — не самый лучший способ проверки доступности служб (гораздо лучше запросить веб-страницу с другого компьютера и убедиться, что в полученной странице есть тег закрытия </html>).

Реализацию этих усовершенствований мы возлагаем на ваши плечи, дорогой читатель. Каждая ситуация уникальна по своей природе, поэтому необходимо самостоятельно определить технологию, наилучшим образом сочетаемую с имеющимися в вашем распоряжении утилитами. В конце концов, в этом и состоит разработка хакерских приемов.

Дополнительная информация:

- пакет Fake от проекта «широкодоступной системы Linux» (High Availability Linux) по адресу *http://www.linux-ha.org/failover/*;
- *«Распределение серверной загрузки при помощи циклического DNS»* **[Совет № 79]**.

СОВЕТ №64 Запуск утилиты ntop для получения сетевой статистики в режиме реального времени

(Следите с помощью утилиты ntop за использованием сети)

Если вас интересует сетевая статистика в режиме реального времени, следует обратиться к утилите ntop. Она представляет собой полнофункциональный анализатор протоколов, обладающий клиентской веб-частью, поддерживающей SSL и имеющей графический интерфейс GD. Утилиту ntop нельзя назвать «легкой» (она может затребовать значительных ресурсов в зависимости от размера сети и объема сетевого трафика), но зато она позволяет составить общее представление и даже выяснить некоторые критичные детали о том, кто с кем общается внутри вашей сети.

Утилиту ntop вначале необходимо запустить под учетной записью пользователя root, чтобы иметь возможность переключить интерфейсы в смешанный режим работы и начать сбор пакетов, после чего с нею сможет работать заданный пользователь. Если хотите, чтобы ntop выполнялась в течение длительного промежутка времени, лучше это сделать на специально выделенном для мониторинга системном блоке, на котором, помимо вышеупомянутой утилиты, запущено всего несколько других служб, предназначенных для обеспечения безопасности и быстродействия.

Расскажем, как быстро запустить утилиту ntop. Во-первых, создайте группу и пользователя для ntop:

```
root@gemini:~# groupadd ntop
root@gemini:~# useradd -c "ntop user" -d /usr/local/etc/ntop \
-s /bin/true -g ntop ntop
```

Затем распакуйте и скомпилируйте ее, как описано в файле инструкции *docs/BUILD-NTOP.txt*. Предположим, у вас имеется распакованное дерево исходных кодов в каталоге */usr/local/src/ntop-2.1.3/*.

Создайте папку, в которой будете хранить базу данных перехваченных утилитой ntop пакетов:

```
root@gemini:~# mkdir /usr/local/etc/ntop
```

Обратите внимание, что владельцем данной папки должен выступать пользователь root, а *не* пользователь ntop.

Если хотите использовать SSL для протокола https (защищенного протокола передачи гипертекстовых данных) вместо стандартного http, необходимо скопировать используемый по умолчанию ключ SSL в папку */usr/local/etc/ntop*:

```
root@gemini:~# cp /usr/local/src/ntop-2.1.3/ntop/*pem /usr/local/etc/ntop
```

Учтите, что стандартный ключ SSL генерируется с некорректным именем узла для сервера. Если хотите научиться создавать собственные ключи, подписанные вашим собственным Центром сертификации, и избавиться от докучливых предупреждений SSL в браузере, прочтите разделы *«Генерирование сертификатов SSL и запросов на их подпись»* **[Совет № 93]** и *«Создание собственного Центра сертификации»* **[Совет № 94]**.

Инициализируйте базы данных ntop и задайте для доступа к ним пароли администратора:

```
root@gemini:~# ntop -A -u ntop -P /usr/local/etc/ntop
21/Sep/2002 20:30:23 Initializing GDBM...
21/Sep/2002 20:30:23 Started thread (1026) for network packet analyzer.
```

```
21/Sep/2002 20:30:23 Started thread (2051) for idle hosts detection.
21/Sep/2002 20:30:23 Started thread (3076) for DNS address resolution.
21/Sep/2002 20:30:23 Started thread (4101) for address purge.

Please enter the password for the admin user:
Please enter the password again:
21/Sep/2002 20:30:29 Admin user password has been set.
```

И наконец, запустите утилиту ntop в качестве демона, а затем — сервер SSL на выбранном номере порта (4242, к примеру):

```
root@gemini:~# ntop -u ntop -P /usr/local/etc/ntop -W4242 -d
```

По умолчанию, утилита ntop осуществляет запуск стандартного сервера http, работающего с портом 3000. Следует тщательно обдумать вопрос блокировки доступа к этим портам на вашем брандмауэре, либо воспользоваться правилами для iptables в командной строке, как описано в разделе *«Создание брандмауэра из командной строки любого сервера»* **[Совет № 45]**. Позвольте утилите ntop поработать некоторое время, а затем подключитесь к адресу *http://your.server.here:4242/*. С помощью ntop можно узнавать различные подробности о трафике сети, как показано на рисунке 5-1.

Info about host nocat.net

IP Address		208.201.239.5 [unicast]
First/Last Seen		09/21/02 21:27:16 - 09/21/02 21:35:20 [8:04]
Domain		net
Last MAC Address/Router		00:01:30:B8:23:D0
Host Location		Remote (outside specified/local subnet)
IP TTL (Time to Live)		54:54 [~10 hop(s)]
Total Data Sent		110.8 KB/1,159 Pkts/0 Retran. Pkts [0%]
Broadcast Pkts Sent		0 Pkts
Data Sent Stats		Local (100 %)
IP vs. Non-IP Sent		IP (100 %)
Total Data Rcvd		384.4 KB/1,372 Pkts/0 Retran. Pkts [0%]
Data Rcvd Stats		Local (100 %)
IP vs. Non-IP Rcvd		IP (100 %)
Sent vs. Rcvd Pkts	Sent (45.8 %)	Rcvd (54.2 %)
Sent vs. Rcvd Data	Sent (22.4 %)	Rcvd (77.6 %)
Further Host Information		[Whois]

Host Traffic Stats

Time	Tot. Traffic Sent	% Traffic Sent	Tot. Traffic Rcvd	% Traffic Rcvd
Midnight - 1AM	0	0.0 %	0	0.0 %
1AM - 2AM	0	0.0 %	0	0.0 %
2AM - 3AM	0	0.0 %	0	0.0 %
3AM - 4AM	0	0.0 %	0	0.0 %
4AM - 5AM	0	0.0 %	0	0.0 %
5AM - 6AM	0	0.0 %	0	0.0 %
6AM - 7AM	0	0.0 %	0	0.0 %

Рис. 5-1. Утилита ntop генерирует статистические графики по сетевому трафику в режиме реального времени

Можно также просматривать эту статистику в графическом виде, удобном для анализа и распечатки, в качестве круговых диаграмм, как показано на рисунке 5-2.

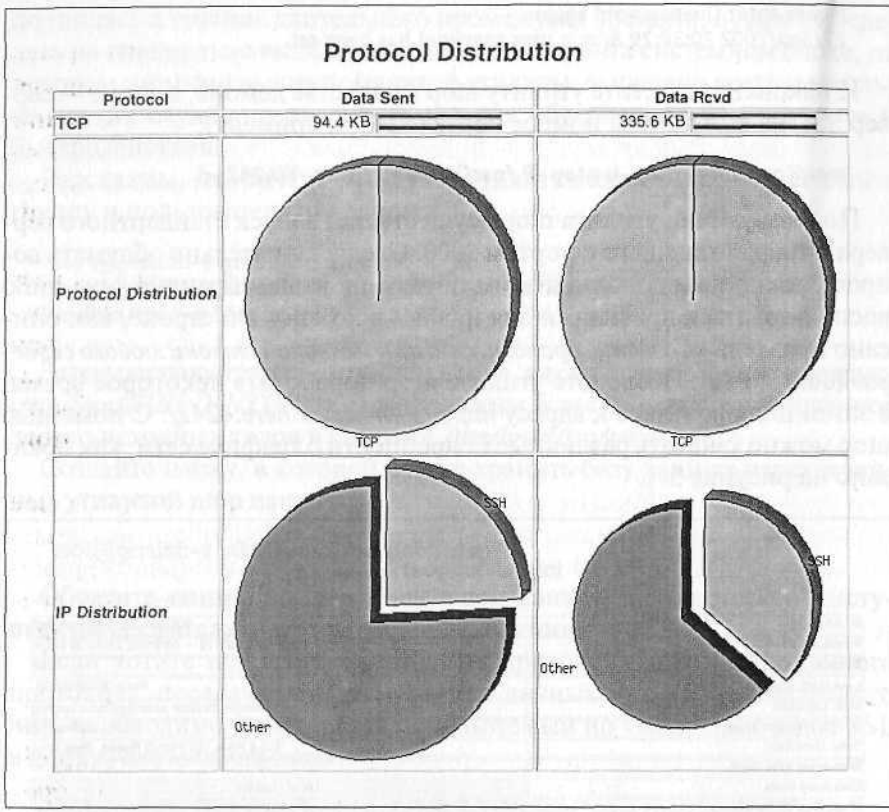

Рис. 5-2. Утилита ntop позволяет узнавать подробности об использовании сети отдельными клиентами

Тогда как утилиты, подобные tcpdump и ethereal, позволяют вести детальный интерактивный анализ сетевого трафика, утилита ntop предоставляет массу ценной статистической информации в виде симпатичного и простого для использования веб-интерфейса. При правильной установке и блокировании несанкционированного доступа ntop может стать вашим любимым средством сетевого анализа.

Дополнительная информация:

- *http://www.ntop.org/*;
- *http://www-serra.unipi.it/~ntop/ntop.html.*

СОВЕТ №65 Мониторинг веб-трафика в режиме реального времени при помощи сценария httptop

(Проследите с помощью httptop, откуда поступает наибольшее количество обращений в секунду)

Расширенные пакеты анализа журналов http, например analog, позволяют получать весьма подробные отчеты о том, кто и какой информацией интересуется на вашем веб-сайте. К сожалению, выполнение задач по обработке журналов может потребовать много времени на загруженном сервере, в результате чего будете иметь ретроспективу событий, лишаясь при этом возможности выяснить, что же происходит прямо *сейчас*.

Другой подход заключается в записи всей веб-активности в базу данных, к которой можно посылать запросы в режиме реального времени. Этот метод позволяет получать мгновенный отчет о текущем состоянии, если, конечно, база данных и веб-сервер обладают достаточной вычислительной мощностью, чтобы выдерживать дополнительную нагрузку. На чрезвычайно загруженных сайтах делать этого не рекомендуется.

Неким промежуточным подходом можно считать использование сценариев на языке Perl, в данном случае сценария httptop, выдающего информацию подобно программе top в режиме реального времени, но без дополнительной нагрузки на сервер. При запуске этого сценария необходимо указать путь к файлу журнала access_log. Наберите в командной строке следующую команду:

```
$ httptop -f combined /usr/local/apache/logs/access_log
```

На экран выводится список, обновляемый каждые несколько секунд и сортируемый по количеству обращений в секунду. Как и в программе top, нажатие ? вызовет вывод информации о списке доступных ключей. Этот сценарий обладает рядом специальных возможностей, таких как сортировка по последнему обращению либо общему количеству обращений и, опционально, отображение URL, доменов, запрошенных URI либо адресов удаленных хостов, с которых поступили запросы.

Если имеются виртуальные узлы сети (VirtualHosts) и требуется одновременно запустить на всех сценарий httptop, воспользуйтесь следующей командой (эту строку необходимо добавить в каждую запись VirtualHost):

```
CustomLog /usr/local/apache/logs/combined-log vhost
```

При этом не забудьте задать где-то в общих настройках файл журнала (на экране все это должно уместиться в одну строку):

```
LogFormat "%v %h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"{User-Agent}i\"" vhost
```

Теперь можно следить за активностью всех узлов сети одновременно при помощи такой команды:

```
$ httptop -f vhost /usr/local/apache/logs/combined-log
```

Учтите, что для работы этого сценария на языке Perl в системе может потребоваться установка модулей Time::HiRes и File::Tail. Если вы не любите набирать программный код вручную, сценарий httptop (как и большинство других примеров, рассматриваемых в данной книге) доступен для загрузки с сайта *examples.oreilly.com*. Точные ссылки указаны во введении.

Листинг: httptop

```
#!/usr/bin/perl -w
#
# $Id: httptop,v 1.1 2002/06/27 00:25:39 rob Exp $

=head1 NAME

httptop – display top(1)-like per-client HTTP access stats
# Статистика доступа HTTP по клиентам

=head1 SYNOPSIS

httptop [-f <format>] [-r <refresh_secs>] [-b <backtrack_lines>] <logdir |
path_to_log>

=cut

use Time::HiRes qw( time );
use File::Tail ();
use Term::ReadKey;
use Getopt::Std;

use strict;

### Стандартные значения, которые могут быть изменены.

my $Update    = 2; # период обновления: через каждые n секунд
my $Backtrack = 250; # Возвратиться на n строк при запуске
my @Paths = qw(
%
/title/%/logs/access_log
/var/log/httpd/%/access_log
/usr/local/apache/logs/%/access_log
);

my $Log_Format  = "combined";
my %Log_Fields = (
```

```
combined => [qw/ Host x x Time URI Response x Referer Client /],
vhost => [qw/ VHost Host x x Time URI Response x Referer Client /]
);

### Константы и другие параметры.
### На этом разделе можно не задерживаться.

my $Version = "0.4.1";

sub by_hits_per () { $b->{Rate} <=> $a->{Rate} }
sub by_total () { $b->{Total} <=> $a->{Total} }
sub by_age () { $a->{Last} <=> $b->{Last} }

my $last_field = "Client";
my $index = "Host";
my $show_help = 0;

my $order = \&by_hits_per;
my $Help = "htlwufd?q";
my %Keys = (
h => [ "Сортировка по частоте обращений / секундам"  => sub { $order =
\&by_hits_per } ],
t => [ "Сортировка по общему числу обращений" => sub { $order = \&by_total } ],
l => [ "Сортировка по последним обращениям" => sub { $order = \&by_age } ],
w => [ "Показать удаленный узел сети" => sub { $index = "Host" } ],
u => [ "Показать запрошенный URI" => sub { $index = "URI" } ],
f => [ "Показать ссылающийся URL" => sub { $index = "Referer" } ],
d => [ "Показать ссылающийся домен" => sub { $index = "Domain" } ],
'?' => [ "Помощь" => sub { $show_help++ } ],
q => [ "Выход" => sub { exit } ]
);

my @Display_Fields = qw/ Host Date URI Response Client Referer Domain /;
my @Record_Fields = qw/ Host URI Referer Domain /;
my $Max_Index_Width = 50;
my $Initial_TTL = 50;

my @Months = qw/ Jan Feb Mar Apr May Jun Jul Aug Sep Nov Dec /;
my %Term = (
HOME => "\033[H",
CLS => "\033[2J",
START_TITLE => "\033]0;", # для xterms и т.п.
END_TITLE => "\007",
START_RV => "\033[7m",
END_RV => "\033[m"
);

my ( %hist, %opt, $spec );

$SIG{INT} = sub { exit };
END { ReadMode 0 };

### Подфункции.

sub refresh_output
```

```
{
my ( $cols, $rows ) = GetTerminalSize;
my $show = $rows – 3;
my $count = $show;
my $now = (shift || time);

for my $type ( values %hist ) {
for my $peer ( values %$type ) {
# if ( —$peer->{_Ttl} > 0 ) {
my $delta = $now – $peer->{Start};
if ( $delta >= 1 ) {
$peer->{ Rate } = $peer->{ Total } / $delta;
} else {
$peer->{ Rate } = 0
}

$peer->{ Last } = int( $now – $peer->{ Date } );
# } else {
# delete $type->{$peer}
# }
}
}

$count = scalar( values %{$hist{$index}} ) – 1 if $show >= scalar values
%{$hist{$index}};
my @list = ( sort $order values %{$hist{$index}} )[ 0 .. $count ];

my $first = 0;
$first = ( $first <= $_ ? $_ + 1 : $first ) for map { $_ ? length($_->{$index}) : 0 }
@list;
$first = $Max_Index_Width if $Max_Index_Width < $first;

print $Term{START_TITLE}, "Monitoring $spec at: ", scalar localtime,
$Term{END_TITLE} if $ENV{TERM} eq "xterm"; # УЖАСНО!!!

my $help = "Help/?";
my $head = sprintf( "%-${first}s %6s %4s %4s %s (%d total)",
$index, qw{ Hits/s Tot Last }, $last_field,
scalar keys %{$hist{$index}}
);

#
# Усечение строки состояния в случае необходимости
#
$head = substr($head, 0, ($cols – length($help)));
print @Term{"HOME", "START_RV"}, $head, " " x ($cols – length($head) –
length($help)), $help, $Term{END_RV}, "\n";

for ( @list ) {
# $_->{_Ttl}++;

my $line = sprintf( "%-${first}s %6.3f %4d %3d %s",
substr( $_->{$index}, 0, $Max_Index_Width ), @$_{(qw{ Rate Total Last },
$last_field)} );
if ( length($line) > $cols ) {
```

```
substr( $line, $cols - 1 ) = "";
} else {
$line .= " " x ($cols - length($line));
}
print $line, "\n";
}

print " " x $cols, "\n" while $count++ < $show;
}

sub process_line
{
my $line = shift;
my $now = ( shift || time );
my %hit;

chomp $line;
@hit{@{$Log_Fields{$Log_Format}}} = grep( $_, split( /"([^"]+)"|\[([^]]+)\]|\s/o,
$line ) );

$hit{ URI } =~ s/HTTP\/1\S+//gos;

$hit{ Referer } = "<unknown>" if not $hit{Referer} or $hit{Referer} eq "-";
( $hit{Domain} = $hit{Referer} ) =~ s#^\w+://([^/]+).*$#$1#os;

$hit{ Client } ||= "<none>";
$hit{ Client } =~ s/Mozilla\/[\w.]+ \(compatible; //(/gos;
$hit{ Client } =~ s/[^\x20-\x7f]//gos;

# Если $now отрицательно, пытаемся определить, как давно производилось
# обращение на основе метки даты/времени.
if ( $now < 0 ) {
my @hit_t = ( split( m![:/\s]!o, $hit{ Time } ))[ 0 .. 5 ];
my @now_t = ( localtime )[ 3, 4, 5, 2, 1, 0 ];
my @mag = ( 3600, 60, 1 );

# Не страшно, даже если обращение не было должным образом обработано
# либо сегодня вообще не было обращений.
return unless $hit_t[2] == ( $now_t[2] + 1900 )
and $hit_t[1] eq $Months[ $now_t[1] ]
and $hit_t[0] == $now_t[0];

splice( @hit_t, 0, 3 );
splice( @now_t, 0, 3 );

# Вернемся назад, ко времени обращения в UNIX.
$now = time;
$now -= (shift( @now_t ) - shift( @hit_t )) * $_ for ( 3600, 60, 1 );
}

$hit{ Date } = $now;

for my $field ( @Record_Fields ) {
my $peer = ( $hist{$field}{$hit{$field}} ||= { Start => $now, _Ttl => $Initial_TTL } );
@$peer{ @Display_Fields } = @hit{ @Display_Fields };
```

```
$peer->{ Total }++;
}
}

sub display_help {
my $msg = "httptop v.$Version";
print @Term{qw/ HOME CLS START_RV /}, $msg, $Term{END_RV}, "\n\n";
print " " x 4, $_, " " x 8, $Keys{$_}[0], "\n" for ( split "", $Help );
print "\nPress any key to continue.\n";
}

### Инициализация.

getopt( 'frb' => \%opt );

$Backtrack = $opt{b} if $opt{b};
$Update  = $opt{r} if $opt{r};
$Log_Format = $opt{f} if $opt{f};
$spec = $ARGV[0];

die <<End unless $spec and $Log_Fields{$Log_Format};
Usage: $0 [-f <format>] [-r <refresh_secs>] [-b <backtrack_lines>] <logdir |
path_to_log>
Valid formats are: @{[ join ", ", keys %Log_Fields ]}.
End

for ( @Paths ) {
last if -r $spec;
( $spec = $_ ) =~ s/%/$ARGV[0]/gos;
}
die "No access_log $ARGV[0] found.\n" unless -r $spec;

my $file = File::Tail->new(
name => $spec,
interval => $Update / 2,
maxinterval => $Update,
tail => $Backtrack,
nowait => 1
) or die "$spec: $!";

my $last_update = time;
my ( $line, $now );

# Откат.
while ( $Backtrack— > 0 ) {
last unless $line = $file->read;
process_line( $line, -1 );
}
$file->nowait( 0 );

ReadMode 4;  # Echo off.
print @Term{"HOME", "CLS"}; # Home & clear.
refresh_output;

### Основной цикл.
```

```
while (defined( $line = $file->read )) {
$now = time;

process_line( $line, $now );

while ( $line = lc ReadKey(-1) ) {
$show_help = 0 if $show_help;
$Keys{$line}[1]->() if $Keys{$line};
}

if ( $show_help == 1 ) {
display_help;
$show_help++; # Don't display help again.
} elsif ( $now - $last_update > $Update and not $show_help ) {
$last_update = $now;
refresh_output( $now );
}
}

__END__

=head1 ОПИСАНИЕ

Сценарий httptop написан в качестве эквивалента top (1) - для оценки активности
httpd.

При вызове сценария httptop необходимо указывать путь к журналу Apache
access_log либо альтернативную строку, уникально идентифицирующую папку,
в которую помещен файл access_log. Каталоги, в которых будет осуществляться
поиск, могут быть заданы на источнике.

Сценарий httptop обладает ограниченной гибкостью при работе с журналами раз-
личных форматов. Запустите 'httptop' без параметров, чтобы узнать список до-
ступных форматов.

Во время выполнения сценария httptop можно получить список терминальных
команд, нажав '?'. Как и для команды top(1), выполнение завершается при нажа-
тии клавиши 'q'.

=head1 ОШИБКИ и другое

Сценарий httptop даже не пытается прочесть файл httpd.conf, чтобы выяснить, где
располагаются файлы журналов либо какой они имеют формат, так что полагаться
на надежность данного сценария нельзя.

Настройку обновления, отката и путей к файлам журнала можно было организо-
вать намного проще.

Сценарий httptop должен предоставлять доступ к суммарной статистике по всему
файлу журнала, подобно uptime(1).

=head1 АВТОР

Schuyler Erle (schuyler@oreilly.com), с предложениями (и некоторыми усовершен-
ствованиями) от Rob Flickenger и Peter Wiggin.
```

```
=head1 ЛИЦЕНЗИЯ

Данное ПО доступно на тех же правах, что и сам язык perl. Исправления и обновления приветствуются.

=head1 ТРЕБОВАНИЯ К СИСТЕМЕ

File::Tail(3), Time::HiRes(3), Term::ReadKey(3)

=head1 СМОТРИТЕ ТАКЖЕ

perl(1), top(1), httpd(8)
```

Защищенная оболочка SSH

Советы № 66-71

При рассмотрении в ходе подготовки этой книги различных методов стало очевидно, что тема защищенной оболочки ssh требует отдельной главы. Такое инструментальное средство, как оболочка ssh, предоставляет в наше распоряжение чрезвычайно гибкие и достаточно защищенные с помощью шифрования методы передачи потоков данных между компьютерами сети. Поскольку командная строка систем Linux обычно выполняет считывание данных из файлов и каналов (pipelines) и запись в них, ssh позволяет передавать данные по сети так, как если бы передача информации осуществлялась внутри локальной системы. Причем все это воплощается быстрым, безопасным и интуитивно понятным способом, что подходит для реализации ряда очень интересных (и мощных) приемов.

Для ОС Linux доступно несколько версий ssh. Предполагается, что для выполнения примеров данной главы будет использоваться OpenSSH версии 3.4p1 либо более поздней. Утилита OpenSSH входит в состав большинства дистрибутивов Linux, но при необходимости ее можно загрузить отдельно по адресу: *http://www.openssh.com/*.

СОВЕТ №66

Быстрый вход в систему при помощи клиентских ключей SSH

(Использование ключей ssh вместо аутентификации с помощью паролей для автоматизации и ускорения процесса подключения к системе)

Если вы работаете администратором на нескольких компьютерах, возможность быстрого подключения к оболочке командной строки заданного сервера приобретает большую важность. Необходимость набирать в командной строке ssh my.server.com (за которой следует пароль) не только надоедает, но и отвлекает.

В любом случае, чем больше сил потрачено на подключение к проблемному компьютеру, тем меньше их останется на решение самой

проблемы. В последних версиях ssh была предложена более безопасная альтернатива бесконечному вводу паролей: обмен открытыми ключами.

Для использования открытых ключей с сервером ssh вначале необходимо сгенерировать пару открытый – закрытый ключ:

```
$ ssh-keygen -t rsa
```

Для генерации ключей DSA воспользуйтесь командой -t dsa, а если применяется Protocol v1, воспользуйтесь командой -t rsa1 (хотя позор вам, если еще используете v1: обновите как можно быстрее до v2!)

После введения вышеупомянутой команды получите результат типа:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/rob/.ssh/id_rsa):
```

Просто нажмите Enter. На просьбу ввести пароль нажмите клавишу Enter дважды (только прочтите перед этим примечание, касающееся безопасности). На экране должно появиться:

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/rob/.ssh/id_rsa.
Your public key has been saved in /home/rob/.ssh/id_rsa.pub.
The Key fingerprint is:
a6:5c:c3:eb:18:94:0b:06:a1:a6:29:58:fa:80:0a:bc rob@localhost
```

В результате будут созданы два файла, *~/.ssh/id_rsa* и *~/.ssh/id_rsa.pub*. Чтобы воспользоваться этой парой ключей на сервере, попробуйте выполнить следующие команды:

```
$ ssh server "mkdir .ssh; chmod 0700 .ssh"
$ scp .ssh/id_rsa.pub имя_сервера:.ssh/authorized_keys2
```

Разумеется, вместо имя_сервера следует подставить конкретное имя. После двойного запроса о пароле наберите ssh имя_сервера, и сможете подключиться автоматически без задания пароля с использованием нового открытого ключа для scp.

Если по каким-то причинам подключение не будет установлено, проверьте разрешения файловой системы в *~/.ssh/** и *имя_сервера:~/.ssh/**. Ваш закрытый ключ – id_rsa – должен иметь разрешения 0600 и существовать только на локальной системе, а для всех остальных файлов должны быть установлены разрешения 0655 и выше.

Теперь можете запускать команду ssh имя_сервера быстро и с минимумом усилий. А можно ли еще больше ускорить подключение к компьютерам, к которым необходимо часто обращаться? Чтобы узнать это, прочтите раздел *«Подключение через ssh в турборежиме»* **[Совет № 67]**.

Вопросы безопасности

Некоторые рассматривают использование открытых ключей как потенциальный риск безопасности. Ведь достаточно, чтобы кто-то украл копию закрытого ключа для получения доступа к серверам! Да, это так, но это же справедливо и для паролей.

Спросите себя, сколько раз в день набираете пароль, чтобы получить доступ из оболочки командной строки к компьютеру или с помощью scp к файлу? Как часто применяете один и тот же пароль на нескольких (а то и всех) компьютерах? Использовали ли вы этот же пароль сомнительным способом (например, на веб-сайте, компьютере с не самым новым программным обеспечением либо в клиенте ssh на системе, которую не можете контролировать напрямую)? Если что-то из этого списка звучит знакомо, учтите, что использование в тех же ситуациях ключа ssh делает для злоумышленника практически невозможным получение несанкционированного доступа к вашим системам, если, конечно, секретный ключ хранится надежно.

Дополнительная информация:

- SSH: Полное руководство (O'Reilly);
- *«Подключение через ssh в турборежиме»* **[Совет № 67]**;
- *«Запуск агента ssh с графическим интерфейсом»* **[Совет № 69]**.

СОВЕТ №67

Подключение через ssh в турборежиме

(Быстрое подключение к системе из командной строки)

Предыдущий совет дает решение только наполовину! Даже имея клиентские ключи, вы по-прежнему вынуждены каждый раз набирать в командной строке ssh имя_сервера при необходимости подключения к ssh. В мрачные и опасные времена rsh существовала одна скрытая функция, которая пока не перенесена в ssh. В rsh можно было воспользоваться символической ссылкой */usr/bin/rsh* на файл, который назывался так же, как сервер, и утилита rsh оказывалась достаточно умной, чтобы понять, что если ее вызывают не под именем rsh, то она должна запуститься под любым именем, с которым была связана.

Реализация этого в оболочке выглядит достаточно тривиально. Создайте файл под именем ssh-to и поместите в него две строки:

```
#!/bin/sh
ssh `basename $0` $*
```

Параметр basename $0 заключен в обратные одинарные кавычки. Теперь поместите этот файл в директорию, указанную в переменной

PATH (если папка *~/bin* не существует или не включена в переменную PATH, немедленно исправьте ситуацию), а затем настройте символические ссылки для всех серверов:

```
$ cd bin
$ ln -s ssh-to server1
$ ln -s ssh-to server2
$ ln -s ssh-to server3
```

Теперь, чтобы подключиться к server1 через ssh (уже должны быть скопированы открытые ключи, как было описано в предыдущем разделе), достаточно набрать в командной строке **server1**, и вы окажетесь в оболочке server1, не набирая ни пароль, ни строку ssh... Символы $* в конце позволяют выполнять произвольные команды из одной и той же командной строки (не порождая новой оболочки):

```
server1 uptime
```

В результате будут показаны время работы системы, количество пользователей, средняя загрузка сервера, а затем произведен выход. Внесите эту команду внутрь цикла for и примените к списку серверов, чтобы выяснить состояние каждой своей системы.

Изложенный способ использования ssh является наиболее быстрым, причем можно еще больше сократить свой труд по набору команд на клавиатуре, задав для выполняемых заданий однобуквенные псевдонимы. В этом случае вы поступите как истинный хакер; правда, обслуживание при этом станет практически невозможным, поэтому ударяться в такие крайности не стоит, хотя это и производит впечатление на многих:

```
$ alias a='ssh alice'
$ alias b='ssh bob'
$ alias c='ssh eve'
...
```

В любом случае, можно сказать, что ssh обладает рядом параметров, превращающих ее в гибкий инструмент безопасной навигации между любым количеством серверов. Теперь, если найдете способ войти в систему и настроить ее под себя, этот совет можно будет действительно назвать хакерским.

Дополнительная информация:

- *«Быстрый вход в систему при помощи клиентских ключей ssh»* **[Совет № 66]**;
- *«Эффективное использование агента ssh»* **[Совет № 68]**.

СОВЕТ №68

Эффективное использование агента ssh

(Применение агента ssh для автоматического управления клиентскими ключами)

Агент ssh представляет собой весьма удобный компонент системы ssh, позволяющий управлять закрытыми ключами, передавая мандаты по первому требованию.

В оперативном руководстве по агенту говорится следующее:

> Агент ssh представляет собой программу для хранения закрытых ключей, используемых во время аутентификации открытого ключа (RSA, DSA). Идея заключается в том, что агент ssh стартует в начале X-сессии либо в начале процедуры входа в систему, а уже все остальные окна или программы запускаются в качестве клиентов агента ssh. Агент легко может быть найден с помощью переменных среды и автоматически использован в целях аутентификации при подключении к другим компьютерам при помощи ssh(1).

На практике это означает, что активный агент, при условии надлежащих настроек клиентов ssh, позволяет подключаться с помощью ssh сразу к нескольким компьютерам без необходимости отправлять на них копии закрытого ключа (либо набирать пароль при каждом подключении).

Предположим, существует ключ авторизации ssh (см. раздел *«Быстрый вход в систему при помощи клиентских ключей ssh»* **[Совет № 66]**), установленный на компьютерах homer, bart и lisa. Подключение с помощью ssh к любому из вышеперечисленных компьютеров с локальной системы не должно вызвать каких-либо проблем:

```
rob@caligula:~$ ssh homer
rob@homer:~$ exit
logout
Connection to homer.oreillynet.com closed.
rob@caligula:~$ ssh bart
rob@bart:~$ exit
logout
Connection to bart.oreillynet.com closed.
rob@caligula:~$ ssh lisa
rob@lisa:~$ exit
```

Но что произойдет, если попытаться подключиться с помощью ssh с компьютера homer к компьютеру bart:

```
rob@caligula:~$ ssh homer
rob@homer:~$ ssh bart
rob@bart's password:
```

Здесь на помощь придет агент ssh. Вместо того чтобы подвергать ненужному риску разглашения свой закрытый ключ, размещая его копии на всех серверах, запустите агент ssh на локальной машине:

```
rob@caligula:~$ eval `ssh-agent`
Agent pid 8450
```

Затем добавьте свои ключи ssh по умолчанию с помощью команды ssh-add:

```
rob@caligula:~$ ssh-add
Identity added: /home/rob/.ssh/id_rsa (/home/rob/.ssh/id_rsa)
Identity added: /home/rob/.ssh/id_dsa (/home/rob/.ssh/id_dsa)
Identity added: /home/rob/.ssh/identity (rob@caligula)
```

Следует также проверить, что компьютеры homer, bart и lisa настроены на перенаправление запросов агента. Обычно по умолчанию задаются именно такие настройки, убедиться в этом можно с помощью следующей строки:

```
ForwardAgent Yes
```

в файле *~/.ssh/config* или */usr/local/etc/ssh_config*. Кроме того, можно задать эти настройки при помощи параметра -A из командной строки.

Теперь, подключив компьютер homer напрямую к компьютеру bart с помощью ssh, homer вначале спросит агента о доступных полномочиях. Аналогично, подключив через ssh компьютер bart к компьютеру lisa, система проведет проверку полномочий компьютера bart на компьютере homer, который перенаправит запрос назад агенту. Таким образом, можно легко переключаться с одного компьютера на другой:

```
rob@caligula:~$ ssh homer
rob@homer:~$ ssh bart
rob@bart:~$ ssh lisa
rob@lisa:~$
```

Итак, в вашем распоряжении имеется очень простой способ сетевой навигации, позволяющий избежать беспокойства по поводу закрытых ключей ssh. Кроме того, он позволяет облегчить при помощи scp копирование файлов между компьютерами.

Но что, если не удастся запустить агент в первую очередь? Такое случается, если вход в систему осуществляется через графическое приглашение XDM либо запущена ОС X. Не бойтесь. Чтобы узнать это, прочтите раздел *«Запуск агента ssh в графическом интерфейсе пользователя»* **[Совет № 69]**.

Разумеется, еще быстрее подключиться к системе через ssh без ущерба для безопасности невозможно. Или все-таки возможно? Если вас интересует правильный ответ, обратитесь к разделу *«Подключение через ssh в турборежиме»* **[Совет № 67]**.

СОВЕТ №69 Запуск агента ssh в графическом интерфейсе пользователя

(Запускайте агент ssh в оконной среде)

Метод использования агента ssh, описанный в предыдущем разделе, хорошо работает в случае, когда запущен до загрузки графической оболочки. Один из способов реализации заключается в запуске агента на стадии входа в систему, в *~/.bash_login* или в *~/.login*, если запущена оболочка tcsh. Но такой вариант решения едва ли можно назвать оптимальным при использовании в ключах парольных фраз. Оконная среда не запустится до тех пор, пока не введены ключевые пароли после входа в систему. Есть еще один побочный эффект: если по какой-то причине работа агента ssh завершится досрочно, вам придется выйти из системы X, запустить агент заново и снова войти в систему. А в некоторых операционных системах, например, OC X, вообще нельзя запустить команды до загрузки графической оболочки.

Вместо того чтобы впустую запускать агент для каждого открываемого окна или копировать и вставлять настройки окружения между ними, добавьте следующий программный код в файл */.profile*:

```
if [ -f ~/.agent.env ]; then
. ~/.agent.env -s > /dev/null

if ! kill -0 $SSH_AGENT_PID > /dev/null 2>&1; then
echo "Stale agent file found. Spawning new agent..."
# Найден старый файл агента. Создаем нового агента...
eval `ssh-agent -s | tee ~/.agent.env`
ssh-add
fi
else
echo "Starting ssh-agent..."
# Запускаем нового агента
eval `ssh-agent -s | tee ~/.agent.env`
ssh-add
fi
```

Таким образом, будет создан файл *~/.agent.env*, причем окружение будет указывать на запущенного в данный момент агента ssh. Если он завершит вдруг свою работу, открытие нового окна терминала приведет к автоматическому запуску нового агента, что добавит для вас новые ключи, а все последующие окна терминалов будут обладать общим доступом к нему.

Теперь у нас имеется один-единственный, но при необходимости запускаемый повторно агент ssh.

Дополнительная информация:

- *«Быстрый вход в систему при помощи клиентских ключей ssh»* **[Совет № 66]**;
- *«Подключение через ssh в турборежиме»* **[Совет № 67]**.

СОВЕТ №70 Запуск X через ssh

(Простой и безопасный удаленный запуск приложений X11 через ssh)

Утилита ssh идеально подходит для перенаправления трафика X11. Если это перенаправление разрешено сервером ssh, к которому вы подключены, то запустить приложение ОС X чрезвычайно просто:

```
rob@florian:~$ ssh -X catlin
Last login: Thu Sep 5 22:59:25 from florian.rob.nocat
Linux 2.4.18

rob@catlin:~$ xeyes &
[1] 12748
rob@catlin:~$
```

Если на локальной системе запущена ОС X, на рабочий стол будет выведено окно программы xeyes. В действительности, выполнение самой программы будет происходить на компьютере catlin, в системе, к которой вы подключены в данный момент. Весь трафик X11 передается в зашифрованном виде через удаленное подключение ssh, отображаемое на локальной системе.

При этом вся работа ложится на плечи утилиты ssh, которая настроит локальный прокси-сервер X11:

```
rob@catlin:~$ echo $DISPLAY
catlin:10.0
```

Обычно по умолчанию перенаправление X11 в openssh отключено. Чтобы включить его, необходимо добавить в файл *sshd_config* следующую строку, после чего перезапустить sshd:

```
X11Forwarding yes
```

Так как программу xeyes нельзя назвать самым полезным примером использования подобной функции, мы приведем несколько более интересных примеров:

ethereal

Выполняет перехват и визуальный анализ пакетов на сервере в *co/lo*

vnc

Принимает на удаленном рабочем столе X команды, поступившие с локального терминала, так, как если бы вы сами сидели за консолью.

gkrellm

Отображает системный статус сервера (или даже нескольких серверов одновременно) в графическом виде.

Чтобы настроить автоматическое перенаправление трафика через ssh, задайте в файле *~/.ssh/config* следующие настройки:

```
ForwardX11 yes
```

Включив этот параметр, больше не придется прибегать к помощи переключателя -X. Этот метод в сочетании со сценарием ssh-to (см. раздел *«Подключение через ssh в турборежиме»* **[Совет № 67]**) обеспечивает самый быстрый способ безопасного удаленного запуска заданий X11:

```
rob#florian:~$ catlin ethereal &
```

В результате тут же будет запущена сессия ethereal на компьютере catlin, шифруемая при помощи ssh. Отбой подключения ssh, скажем, при помощи ~, мгновенно прервет выполнение всех активных приложений X. Но если понадобится временно приостановить их выполнение, прервите сессию ssh при помощи ~^Z, а затем в нужный момент, восстановите ее с помощью fg.

Дополнительная информация:

- сетевой монитор Ethereal: *http://www.ethereal.com/*;
- VNC (свободно распространяемый клиент удаленного рабочего стола для ОС X): *http://www.uk.research.att.com/vnc/*;
- системный монитор gkrellm: *http://www.gkrellm.net/*.

СОВЕТ №71

Переназначение портов в ssh

(Безопасная передача сетевого трафика на произвольные порты путем переназначения портов ssh)

Помимо обеспечения удаленного доступа к оболочке командной строки и удаленного выполнения команд, утилита OpenSSH позволяет переназначать произвольные порты TCP на другом конце соединения. Это очень удобно для защиты электронной почты, веб или же любого другого трафика, который должен быть защищен (по крайней мере, по пути на другой конец туннеля).

Модуль ssh осуществляет локальное перенаправление с помощью привязки к локальному порту с последующим шифрованием, отправкой зашифрованных данных на другой конец подключения ssh, а затем расшифровкой полученных данных и их отправкой на заданный порт удаленного узла сети. Инициализировать туннель ssh можно с помощью переключателя -L (сокращение от Local – локальный):

```
root@laptop:~# ssh -f -N -L110:почтовый_сервер:110 -l пользователь
почтовый_сервер
```

Разумеется, вместо параметра ***почтовый_сервер*** необходимо подставить имя почтового сервера либо его IP-адрес, а вместо ***пользователь*** — имя пользователя почтового сервера. Учтите, что для выполнения этого примера необходимо иметь права пользователя root на удаленной системе laptop, поскольку привязка выполняется к порту 110 (POP), требующему соответствующих привилегий. Понадобится также отключить все демоны POP, активные на локальной системе (проверьте файл */etc/inetd.conf*), чтобы они не помешали вашей работе в самый неподходящий момент. Теперь для выполнения шифрования всего POP-трафика настройте свой почтовый клиент на подключение к локальному порту 110. Таким образом можно без проблем общаться с почтовым сервером, как при подключении к нему напрямую, но весь обмен информацией будет происходить в зашифрованном виде.

Переключатель -f позволяет перевести ssh в фоновый режим работы, а -N отменяет реальное выполнение команды на другом конце туннеля, осуществляя только перенаправление. Если сервер ssh поддерживает данную функцию, воспользуйтесь переключателем -C для включения сжатия — это позволит значительно сократить время загрузки почты.

Можете задать столько строк с переключателем -L, сколько необходимо при установке подключения. Для перенаправления исходящего почтового трафика воспользуйтесь следующей командой:

```
root@laptop:~# ssh -f -N -L110:почтовый_сервер:110 -L25:почтовый_сер-
вер:25 \ -l пользователь почтовый_сервер
```

В качестве сервера исходящих сообщений задайте свой локальный компьютер, и весь ваш исходящий почтовый трафик будет шифроваться точно так же, как входящий на почтовом сервере. Функция полезна только в случае привязки почтового сервера ко внутреннему узлу сети, либо вы не доверяете локальной сети — ситуация, обычная для большинства беспроводных сетей. Очевидно, что как только корреспонденция покинет почтовый сервер, она будет передаваться дальше в виде простого текста, если только не использованы утилиты шифрования, такие как pgp или gpg.

Если уже есть подключение к удаленной системе и необходимо быстро переназначить порт, придерживайтесь следующего алгоритма:

1. Нажмите Enter.
2. Наберите ~C.
3. В строке приглашения ssh> наберите строку -L из оболочки командной строки.

Например:

```
rob@catlin:~$
rob@catlin:~$ ~, затем C (эти символы не будут отображаться на экране)
ssh> -L8000:localhost:80
Forwarding port
```

Теперь текущая оболочка, в которой вы находитесь, будет перенаправлять трафик с локального порта 8000 на 80-й порт компьютера catlin, как если бы это подключение было изначальным.

Можете позволить другим (удаленным) клиентам подключаться к переназначенному таким образом порту при помощи переключателя -g. При подключении к удаленному шлюзу, играющему роль транслятора сетевых адресов (NAT) в локальной сети, командная строка должна будет выглядеть так:

```
rob@gateway:~$ ssh -f -g -N -L8000:localhost:80 10.42.4.6
```

С ее помощью все подключения, поступившие с порта 8000 шлюза, будут перенаправлены на порт 80 внутреннего узла сети. Если шлюз имеет действующий адрес в сети интернет, можно подключиться к веб-серверу 10.42.4.6 из всемирной паутины, как если бы вы напрямую работали с портом 8000 шлюза.

И последний момент, о котором следует упомянуть: трафик может перенаправляться не только на локальную систему; можете выбрать любой компьютер, к которому шлюз имеет непосредственный доступ. Например, чтобы перенаправить трафик с локального порта 5150 на веб-сервер, размещенный где-то во внутренней сети, воспользуйтесь следующей командой:

```
rob@remote:~$ ssh -f -N -L5150:intranet.insider.nocat:80 gateway.nocat.net
```

Предположим, что на вашей системе запущен TLD (Top Level DOMAIN — домен верхнего уровня, см. раздел *«Запуск собственного домена верхнего уровня»* **[Совет № 80]**) для *.nocat*, а *gateway.nocat.net* также подключен к локальной сети *.nocat*. При этом весь удаленный трафик, направленный на локальный порт 5150, будет явным образом перенаправляться на *intranet.insider.nocat:80*. Адрес *intranet.insider.nocat* не потребует разрешения адресов DNS при обращении с узла *remote*: поиск этого узла не будет осуществляться до тех пор, пока не будет создано

подключение к узлу *gateway.nocat.net*, после чего поиск будет выполняться самим шлюзом. Для безопасной навигации по сайту с узла *remote* попытайтесь подключиться по адресу *http://localhost:5150/*.

Хотя утилита ssh обладает функциональностью прокси Socks 4, при задании переключателя -D, она не слишком подходит для перенаправления всего сетевого трафика на другой конец туннеля. Обратитесь к разделу *«Туннелирование: инкапсуляция IPIP»* **[Совет № 50]**, где расписан способ использования vtun в содружестве с ssh для перенаправления всего трафика. И обратите внимание на документацию по использованию переключателя -D, поскольку он применяется довольно редко.

Утилиту ssh по праву можно назвать весьма гибким инструментом, обладающим намного большим количеством функций, чем здесь описано. Более подробную информацию об ssh можно получить, прочитав приведенные ниже ссылки.

Дополнительная информация:

- оперативное руководство по ssh (команда man ssh);
- SSH, защищенная оболочка: полное руководство (O'Reilly);
- *«Туннелирование: инкапсуляция IPIP»* **[Совет № 50]**;
- *«Быстрый вход в систему при помощи клиентских ключей ssh»* **[Совет № 66]**;
- *«Запуск вашего собственного домена верхнего уровня»* **[Совет № 80]**.

Написание сценариев

Советы № 72-75

Иногда возникает необходимость выполнения определенных задач, решить которые с помощью одной командной строки невозможно. После того как такая необходимость возникнет у вас хотя бы несколько раз, вы, безусловно, задумаетесь о более удобных способах решения подобных задач (при которых не нужно будет каждый раз набирать много строк программного кода). Именно на этом этапе системные администраторы нередко начинают привлекать к решению проблем лиц, называемых *программистами*.

Данная глава не является кратким курсом по программированию, скорее ее следует рассматривать как набор примеров по прикладному программированию для хакеров. Я пришел к выводу, что наилучший способ научиться программированию заключается в самостоятельном выполнении заданий, а наилучший способ быстро вникнуть в него – это увидеть, как подобные проблемы решают другие люди. Примеры, рассмотренные в данной главе, полезны сами по себе, но еще большую пользу вы сможете извлечь, если воспользуетесь ими в качестве отправной точки для создания ваших собственных сценариев. Даже если вы имеете немалый опыт в написании сценариев, советуем взглянуть на некоторые примеры использования редко применяемых переключателей и языковых возможностей, дабы не пришлось тратить лишних усилий на получение нужного результата .

СОВЕТ №72

Быстрый перенос настроек при помощи сценария movein..sh

(Синхронизация локального окружения со всеми вашими серверами)

Когда вы в течение некоторого времени пользуетесь системой, то, в конце концов, настраиваете ее в соответствии со своими предпочтениями. Как мы говорили в разделе *«Как чувствовать себя комфортно в оболочке командной строки»* **[Совет № 10]**, окружение оболочки ко-

мандной строки представляет собой необычайно гибкое инструментальное средство, которое можно очень тонко настроить в соответствии с вашими требованиями.

Обычно подбор таких «тонких» настроек требует недель (а то и лет работы), при этом подобные усовершенствования охватывают всего несколько файлов: различные файлы настроек окружения (в зависимости от того, имеете ли вы дело с оболочкой входа в систему либо с внутренней оболочкой), настройки редактора, почтовые параметры, настройки mysql, псевдонимы и опции для каждой ситуации и т.д. Когда рабочая среда настроена соответственно вашим требованиям, облегчается как работа с самой системой, так и выполнение требуемых задач.

Увы, все ваши старания сводятся на «нет» при необходимости подключения к удаленной системе. Бывает весьма обидно набрать свою любимую команду и увидеть в ответ bash:xyz:command not found (команда не найдена). Либо запустить утилиту ls и не увидеть удобного цветового форматирования результатов. Я уже не говорю о сотнях других мелких усовершенствований, превративших работу с вашим домашним компьютером в удовольствие, отсутствие которых делает работу с удаленными компьютерами безрадостной рутиной.

Конечно, вы всегда можете скопировать свои настройки вручную, поскольку большинство из них хранятся в файлах шаблонов внутри вашей домашней директории (например, *..bashrc* или *..vimrc*). Однако запомнить все файлы, в которых вы меняли настройки вручную, достаточно сложно, что приводит к классической проблеме синхронизации версий: когда вы что-то меняете локально, вам необходимо также внести изменения во все удаленные копии.

Сделайте свою жизнь легче при помощи следующего очень простого сценария оболочки.

Листинг: movein..sh

```
#!/bin/sh

if [ -z "$1" ]; then
echo "Usage: `basename $0` hostname"
exit
fi

cd ~/.skel
tar zhcf - . | ssh $1 "tar zpvxf -"
```

Назовите сценарий *movein..sh* и запишите его в папку *~/bin*. Теперь создайте директорию под названием *~/.skel*, а в ней – символические ссылки на все файлы, которые должны синхронизироваться с удаленными серверами:

```
rob@caligula:~/.skel$ ls -al
total 12
drwxr-xr-x 6 rob staff 204 Sep 9 20:52 .
drwxr-xr-x 37 rob staff 1258 Sep 9 20:57 ..
lrwxr-xr-x 1 rob staff 11 Sep 9 20:52 ..bash_login -> .../.bash_login
lrwxr-xr-x 1 rob staff 11 Sep 9 20:52 ..bashrc -> .../.bashrc
lrwxr-xr-x 1 rob staff 11 Sep 9 20:52 .my..cnf -> ../.my..cnf
lrwxr-xr-x 1 rob staff 11 Sep 9 20:52 .pinerc -> .../.pinerc
drwxr-xr-x 3 rob staff 102 Sep 9 20:51 ..ssh
lrwxr-xr-x 1 rob staff 9 Sep 9 20:52 ..vimrc -> .../.vimrc
lrwxr-xr-x 1 rob staff 6 Sep 9 21:27 bin -> .../bin
```

Обратите внимание на *~/.skel/.ssh*: это особый случай – не директория и не символическая ссылка. НЕ СОЗДАВАЙТЕ СИМВОЛИЧЕСКУЮ ССЫЛКУ ДЛЯ *~/.ssh* в *~/.skel*! В последнюю очередь вам нужно будет скопировать закрытый ключ ssh на интересующие вас серверы – для этого предназначен агент *ssh* (см. *«Эффективное использование агента ssh»* **[Совет № 68]**). Вместо этого создайте директорию под названием *~/.skel/.ssh* и задайте для нее следующую символическую ссылку:

```
rob@caligula:~/.skel$ cd .ssh
rob@caligula:~/.skel/.ssh$ ls -al
total4
drwxr-xr-x 3 rob staff 102 Sep 9 20:51 ..
drwxr-xr-x 6 rob staff 204 Sep 9 20:52 ..
lrwxr-xr-x 1 rob staff 26 Sep 9 20:51 autorized_keys2 ->
.../../.ssh/id_dsa.pub
```

Эту ссылку мы назовем *autorized_keys2*, и она будет указывать на ваш действующий открытый ключ. Ведь вы используете при установке подключения *ssh* открытые ключи, не так ли? Если нет, прочтите раздел *«Быстрый вход в систему при помощи клиентских ключей ssh»* **[Совет № 66]**.

Теперь, запустив этот сценарий, вы осуществите копирование содержимого папки *~/.skel* на заданный в командной строке узел сети прямиком в свою домашнюю директорию. Флажок h сообщает утилите tar о необходимости копировать не символические ссылки, а сами файлы, на которые они указывают. Если вы внесете изменения в какой-то из файлов этого списка, достаточно будет еще раз запустить сценарий – и все изменения отразятся на удаленных компьютерах.

Включайте директорию *..ssh* вышеупомянутым способом только в том случае, если хотите подключаться к удаленным узлам сети автоматически, не задавая пароли. Если на вашем локальном компьютере ключи хранятся в безопасности, никакого особого риска в том, что ваши дополнительные файлы *autorized_keys2* будут размещены на других компьютерах сети, нет. В этом и заключается преимущество криптографии с использованием открытых ключей.

Дополнительная информация:

- *«Быстрый вход в систему при помощи ключей ssh»* **[Совет № 66]**
- *«Эффективное использование агента ssh»* **[Совет № 68]**
- *«Как чувствовать себя комфортно в оболочке командной строки»* **[Совет № 10]**
- SSL: Защищенная оболочка, полное руководство (O'Reilly)

СОВЕТ №73

Глобальный поиск и замена с помощью языка Perl

(Манипулирование файлами и потоками с произвольными подстановками Perl без помощи сценариев)

Существует целый ряд переключателей, делающих язык Perl очень полезным средством редактирования, запускаемым из командной строки. Выучите эти переключатели, и вы сможете в одну строку уместить магические команды на языке Perl, приводя в замешательство друзей (и ужасая менеджеров проекта).

Первым из этого списка мы рассмотрим переключатель -e. Задавая этот переключатель после строки программного кода, вы заставите его выполняться так, как если бы это был обычный сценарий на языке Perl:

```
rob@catlin:~$ perl -e 'print "Hi, Ma!\n"'
Hi, Ma!
```

Обратите внимание: в команде, умещающейся в одну строку, не нужно использовать завершающий символ ;. Обычно желательно также заключать командную строку в одинарные кавычки, чтобы оболочка не пыталась интерпретировать специальные символы (такие как ! и \).

Следующий переключатель -p кажется более сложным, однако он проявит свою вторую натуру, как только вы начнете его применять. Из документации по Perl:

> **-p** создает вокруг вашей программы следующий цикл, что позволяет многократно проходить этот цикл, используя в качестве аргументов различные имена файлов, подобно **sed**:
>
> ```
> LINE:
> while (<>) {
> ... # здесь находится ваша программа
> } continue {
> print or die "-p destination: $!\n";
> }
> ```

Строка $_ будет автоматически выводиться при каждой итерации. Если воспользуетесь сразу двумя переключателями: -p и -e, получите однострочную команду, которая будет обрабатывать последовательно каждый файл, указанный в командной строке, либо текст, пе-

реданный в STDIN. Рассмотрим в качестве примера более сложную команду cat:

```
rob@catlin:~$ perl -pe 1 /etc/hosts
#
# В файле hosts приведен список соответствий имен узлов сети их адресам
# для подсистемы TCP/IP
#

# Для организации обратной связи
127.0.0.1 localhost
```

1 означает просто код возврата (как если бы ввели 1; в сценарии на языке Perl это эквивалентно команде return 1;). Поскольку данные строки выводятся автоматически, нам вообще не нужно задавать переключатель -e.

Ситуация становится интереснее, если мы попробуем написать программный код для осуществления манипуляций с текущей строкой до того, как она будет выведена на экран. Например, предположим, что вы хотите добавить в строку с localhost имя вашего локального компьютера:

```
rob@catlin:~$ perl -pe 's/localhost/localhost $ENV{HOSTNAME}/' /etc/hosts
#
# В файле hosts приведен список соответствий имен узлов сети их адресам
# для подсистемы TCP/IP
#

# Для организации обратной связи
127.0.0.1 localhost
```

либо желаете некоторым образом изменить настройки inetd:

```
rob@caligula:~$ perl -pe 's/^(\s+)?(telnet|shell|login|exec)/# $2/' \
/etc/inetd.conf
```

В результате содержимое файла */etc/inetd.conf* будет направлено в STDOUT, причем окажутся закомментированными строки вызова служб *telnet, shell, login* и *exec*. Естественно, вы можете перенаправить выходные данные назад в файл, но если нам нужно его только отредактировать, существует лучший способ: переключатель -i.

-i позволяет выполнять интерактивное редактирование файлов. Поэтому чтобы закомментировать все вышеперечисленные строки в файле */etc/inetd..conf*, вы можете попробовать воспользоваться следующей командой:

```
rob@catlin:~# perl -pi -e s/^(\s+)?(telnet|shell|login|exec)/# $2/'
/etc/inetd.conf
```

еще лучше следующий вариант:

```
rob@catlin:~# perl -pi.orig -e s/^(\s+)?(telnet|shell|login|exec)/# $2/'
/etc/inetd.conf
```

Во втором случае перед изменением оригинала будет создана резервная копия файла */etc/inetd..conf* под именем */etc/inetd..conf..orig*. Не забудьте перезапустить inetd, чтобы изменения вступили в силу.

Так же просто одновременно редактировать несколько файлов. Вы можете задать любое количество файлов (либо подстановочных символов) в командной строке:

```
rob@catlin:~# perl -pi.bak -e 's/bgcolor=#ffffff/bgcolor=#000000/i' *.html
```

Результатом выполнения этой команды станет изменение фонового цвета на всех html-страницах текущей директории с белого на черный. Не забудьте завершающий символ i, чтобы при задании цвета регистр символов не различался (и вы могли задавать параметр bgcolor =#FFFFFF или даже BGColor=#FfFfFf).

А что, если у вас в самом разгаре работа над проектом, контролируемым с помощью CVS (Concurrent Versioning System – системы контроля параллельных версий), а вам нужно сменить CVS-сервер и произвести на нем фиксацию изменений? Это очень просто сделать с помощью команды perl -pi -e, если у вас настроена передача результатов выполнения команды find через xargs:

```
schuyler@ganesh:~/nocat$ find -name Root | xargs perl -pi -e
's/cvs.oldserver.com/cvs.newsletter.org/g'
```

Содержимое переменной $CVSROOT будет сброшено, и проверка CVS пройдет как обычно, после чего ваш проект автоматически будет проверен с помощью *cvs..newserver..org..*

Использование языка Perl в командной строке поможет вам «на лету» выполнять многие сложные преобразования. Изучайте и применяйте с умом регулярные выражения, и вы сможете избежать редактирования вручную.

Дополнительная информация:

- Программирование на языке Perl, (O'Reilly)
- perldoc perlrun
- *«Как чувствовать себя комфортно в оболочке командной строки»* **[Совет № 10]**
- *«CVS: Внесение изменений в модуль»* **[Совет № 30]**

СОВЕТ №74 Деление данных на порции произвольной величины (в оболочке bash)

(Использование арифметических возможностей оболочки bash и команды dd для разбиения больших бинарных файлов на меньшие фрагменты)

Ниже приводится пример использования переменных среды и арифметических выражений в оболочке командной строки bash для разделения файлов на фрагменты произвольного размера.

Листинг: mince

```
#!/bin/bash

if [ -z "$2" -o ! -r "$1" ]; then
echo "Usage: mince [file] [chunk size]"
# Использование: mince [имя_файла] [размер_фрагмента]
exit 255
fi

SIZE=`ls -l $1 | tr -s 'a-zA-Z -' ' ' | cut -f 3 -d ' '`
echo "size = $SIZE"

if [ $2 -gt $SIZE ]; then
echo "Your chunk size must be smaller than the file size!"
# Заданный размер фрагмента должен быть меньше, чем размер файла
exit 254
fi

CHUNK=$2
TOTAL=0
PASS=0
while [ $TOTAL -lt $SIZE ]; do
PASS=$((PASS + 1))
echo "Creating $1.$PASS..."
dd conv=noerror if=$1 of=$1.$PASS bs=$CHUNK skip=$((PASS - 1)) count=1 2>
/dev/null
TOTAL=$((TOTAL + CHUNK))
done

echo "Created $PASS chunks out of $1."
# Создано $PASS фрагментов файла $1
```

Обратите внимание: мы воспользовались преимуществами выражения conv=noerror, поскольку последний фрагмент файла, как правило, оказывается меньше, чем заданный размер порции. Использование этой опции в утилите dd заставляет ее продолжать запись битов до тех пор, пока не закончится исходный файл, после чего выполнение команды будет завершено (без выброса ошибки, и, что еще более важно, dd не откажется записывать последний фрагмент).

Этот сценарий может пригодиться для разбивки больших файлов на фрагменты, которые можно было бы уместить на сменные диски (zip, CD-R, DVD, Usenet) перед архивированием. Поскольку мы используем функции игнорирования ошибок утилиты dd, она будет работать с файлами любых размеров, независимо от доступного объема ОЗУ (если вы, конечно, будете разбивать файл на фрагменты разумной величины). Поскольку значение размера блока (bs) будет равняться размеру порции, сценарий будет выполняться достаточно быстро, особенно если размер порции составит более нескольких килобайт.

Полученные порции будут сохранены в виде набора файлов (имена которых будут заканчиваться на символ точки (.)), после которого должен следовать порядковый номер порции в текущем каталоге. Чтобы увидеть отсортированный по номеру список файловых фрагментов, запустите команду ls FILENAME.*. Но что, если при разбивке файла мы получим более девяти частей?

```
ls FILENAME..* | sort -n -t . +2
```

Как теперь заново собрать файл из отдельных порций?

```
cat `ls FILENAME.* | sort -n -t . +2` > FILENAME..complete
```

Вы мне не верите?

```
diff FILENAME FILENAME..complete
```

Разумеется, данная команда подразумевает, что в имени вашего файла содержится только одна разделительная точка. Если же у вас *очень.длинное.имя.файла.с.несколькими.точками*, обратитесь к оперативному руководству по команде sort(1).

СОВЕТ №75 Анализ журналов, выводимых на экран с цветовым форматированием

(Использование цветового форматирования при анализе файлов журнала в окне xterm)

Если вы ловите себя на том, что построчный анализ системных журналов занимает слишком много времени, то следует поразмыслить над использованием утилит для лучшей систематизации своих журналов. Хотя правильно настроенный демон syslog (см. *«Управление службой syslog»* **[Совет № 54]**) позволяет привести файлы журналов в намного более опрятный вид, все-таки ручной анализ многих мегабайт сообщений в файле */var/log/messages* в поисках знакомых шаблонов может отнять слишком много сил.

Точно так же, как применение цветового форматирования в команде ls позволяет различать типы файлов с первого взгляда, ис-

пользование аналогичного форматирования в команде grep позволяет обнаруживать шаблоны среди бесчисленного множества однообразных строк текста. Существует немало приложений для OC X, позволяющих сделать это за вас, но почему бы не облегчить себе просмотр файлов журнала с цветовым форматированием прямо из командной строки?

Сохраните следующий программный код в файл под названием *~/bin/rcg* (Regex Colored Glasses):

```
#!/usr/bin/perl -w
use strict;
use Term::ANSIColor qw(:constants);

my %target = ( );

while (my $arg = shift) {
 my $clr = shift;

 if(($arg =~ /^-/) | (!$clr)) {
 print "Usage: rcg [regex] [color] [regex] [color] ....\n";
 exit;
 }

 #
 # Безобразно написанный (из-за лени) прием
 #
 $target{$arg} = eval($clr);
}

my $rst = RESET;

while(<>) {
 foreach my $x (keys(%target)) {
 s/($x)/$target{$x}$1$rst/g;
 }
 print;
}
```

rcg представляет собой простой фильтр, использующий Term::ANSIColor для раскраски различных шаблонов регулярных выражений (regexs – REGular EXpressions), который можно задать из командной строки. Этот фильтр предназначен для облегчения визуального анализа файлов журнала..

Вам понадобится передать команде rcg четное количество параметров командной строки. При этом нечетные элементы будут определять регулярные выражения, а четные – цвет.

Предположим, вы хотите, чтобы все, что связано со словом sendmail внутри сообщений, выделялось фиолетовым цветом:

```
$ rcg sendmail MAGENTA < /var/log/messages | less -r
```

Применять параметр less -r необязательно, но удобно (тогда в less также будут отображаться заданные цвета, а не ESC-последовательности).

Вы можете использовать произвольные регулярные выражения в роли нечетных элементов:

```
$ rcg '\d+\.\d+\.\d+\.\d+' GREEN < /var/log/maillog
```

либо связать цвета вместе:

```
$ tail -50 /var/log/messages | rcg WARNING "BOLD . YELLOW . ON_RED"
```

Кроме того, в одной командной строке можно задать произвольное количество пар – регулярное выражение/цвет. Здесь нам пригодятся небольшие сценарии оболочки (один для сообщений, другой для журналов брандмауэра, третий для сервера Apache).

Смотрите документацию по Term::ANSIColor для получения полного списка цветов и комбинаций.

Еще некоторые полезные строки:

```
\w+=\S+
```

переменные, такие как TERM=xyz

```
\d+\.\d+\.\d+\.\d+
```

вероятно, IP-адрес

```
^(J|F|M|A|S|O|N|D)\w\w (\d| )\d
```

может быть, дата

```
\d\d:\d\d:\d\d
```

может быть время

```
.*повтор последнего сообщения..*
```

выделяет полужирным белым цветом.

Используйте свое воображение, но учтите: параметры цвета имеют свои значения. Теоретически, многие допустимые выражения на языке Perl могут быть подставлены вместо регулярных выражений или цветов и мы предлагаем нашим читателям потренироваться в этом в качестве упражнений. Конечно, не стоит запускать произвольные строки rcg, если только не вы сами их написали. Учтите также, что цветовое форматирование применяется в произвольном порядке, так что гарантировать отображение в определенном цвете пересекающихся регулярных выражений практически невозможно.

Информационные серверы

Советы № 76-100

ОС Linux представляет собой мощную платформу для построения информационных серверов. В то же время сами по себе информационные системы редко становятся частью операционной системы Linux. Она, как правило, выступает в роли «системы жизнеобеспечения» для более сложных, специальных информационных служб.

В последней главе этой книги мы рассмотрим три приложения, получившие наибольшее распространение. Все они хорошо зарекомендовали себя под управлением ОС Linux и в настоящее время стали основой информационных служб в сети интернет. Служба BIND (разработанная Internet Software Consortium) представляет собой самый распространенный информационный сервер DNS в сети интернет, предоставляющий информацию о соответствии доменных имен IP-адресам, без которой работа всемирной сети была бы невозможна. Для информационных нужд общего плана можно использовать MySQL: облегченную, быстродействующую и масштабируемую базу данных SQL, обслуживающую многие сетевые приложения корпораций. И наконец, когда речь заходит о предоставлении информации конечным пользователям, на первом месте по популярности среди всех веб-серверов на планете стоит сервер Apache. Количество серверов под его управлением превышает суммарное количество веб-серверов под управлением других служб по вполне понятным причинам: он зарекомендовал себя как зрелый, стабильный и быстродействующий программный продукт, полный полезных и интересных возможностей.

Если вас интересует более подробная информация о работе любого из вышеперечисленных программных пакетов, обращайтесь к онлайновой документации отдельно по каждому из них. Эти приложения отвечают за текущее воплощение всемирной паутины, и поэтому они широко и подробно документированы. Кроме того, издательский дом O'Reilly выпустил целый ряд неплохих книг о каждом из трех вышеперечисленных приложений: «DNS и BIND»; «Справочное руководство по MySQL»;

«Apache: полное руководство» и другие. Отличным руководством по работе с MySQL также является книга Пола Дюбуа (New Riders) «MySQL».

В последующих разделах будут рассмотрены некоторые неочевидные на первый взгляд технологии, позволяющие серверам предоставлять информацию требуемым способом. Мы изучим возможности применения некоторых методов в масштабе больших инсталляций, сохраняя при этом обслуживание даже очень сложных сайтов на минимальном уровне.

СОВЕТ №76

Запуск BIND в окружении chroot

(Запускайте службу named с помощью окружения chroot изолированно от остальной части системы)

Подавляющее большинство сети интернет целиком полагается на BIND в плане разрешения имен. Хотя на укрепление потенциальных брешей системы безопасности службы BIND были брошены грандиозные силы, нельзя быть абсолютно уверенным в том, что некоторый программный код не будет использован злонамеренным образом. Чтобы минимизировать возможный ущерб, вызванный удаленной эксплуатацией брешей с правами пользователя root (из-за ошибок переполнения буфера, программных сбоев или неправильных настроек), на многих сайтах запуск службы named организован в окружении chroot. Благодаря этому, даже если процесс named окажется скомпрометированным, это не прибавит свободы действий хакеру. Окружение chroot не всегда бывает непробиваемым, однако оно позволяет значительно осложнить жизнь взломщику системы.

Следующие действия описывают минимальные усилия, прилагаемые, чтобы заставить BIND 9 работать в окружении chroot. Ведь безопасность DNS — большой и сложный вопрос, который должен рассматриваться серьезно. Обратитесь к источникам, указанным в конце данного параграфа, для получения более подробной информации.

Для начала необходимо запустить службу named под учетной записью другого пользователя (не root). Создайте пользователя и группу, которые будут использоваться исключительно для запуска этого процесса:

```
root@gemini:~# groupadd -g 53 named
root@gemini:~# useradd -u 53 -g named -c «chroot BIND user» \
-d /var/named/jail -m named
```

Заставьте процесс named выполняться в окружении chroot в папке */var/named/jail* во время автозагрузки. Необходимо создать минимальный скелет файловой структуры внутри данной директории для обеспечения нормального запуска процесса. Создайте каталог */var* и скопируйте в него свои данные DNS:

```
root@gemini:~# cd ~named
root@gemini:/var/named/jail# mkdir -p var/{run,named}
root@gemini:/var/named/jail# cp -Rav /var/named/data var/named/
```

Если вы выступаете в качестве ведомого узла для произвольного количества зон, тогда процесс должен иметь право на запись в эту директорию, чтобы отслеживать изменения данных на ведомом узле. Создайте специальную папку для хранения данных ведомой системы, независимо от своих обычных файлов данных:

```
root@gemini:/var/named/jail# mkdir var/named/slave
root@gemini:/var/named/jail# chown named.named var/named/slave
```

Создайте папки с именами dev/ и etc/, в которые необходимо скопировать критичные файлы и устройства системы:

```
root@gemini:/var/named/jail# mkdir {dev,etc}
root@gemini:/var/named/jail# cp -av /dev/{null,random} dev/
root@gemini:/var/named/jail# cp -av \
/etc/{localtime,named.conf,rndc.key} etc/
```

Сбросьте владельцев и разрешения для этих папок:

```
root@gemini:/var/named/jail# chown root.root .
root@gemini:/var/named/jail# chmod 0755 .
root@gemini:/var/named/jail# chown named.named var/named/data/
root@gemini:/var/named/jail# chmod 0700 var/named/data
root@gemini:/var/named/jail# chown named.named var/run
```

Если для ведения журналов DNS используется служба syslog, добавьте переключатели, приведенные в следующей командной строке, в ресурсы запуска демона syslogd. Это заставит службу syslog прослушивать сокет, т.к. системная папка */dev/log* недоступна изнутри окружения:

```
syslogd -m 0 -a /var/named/jail/dev/log
```

Если же используются журналы файловой системы, конфигурацию syslogd менять не нужно. Достаточно убедиться, что указанный пользователь может вести запись в файлы журнала и что внутри папки */var/named/jail/* существует ваша папка журналов.

Запустите процесс named, задав имя пользователя, директорию *chroot* и начальный файл конфигурации из командной строки:

```
root@gemini:~# /usr/sbin/named -u named -t /var/named/jail \
-c /etc/named.conf
```

Если служба named не запускается нормально, посмотрите системные файлы */var/log/syslog* или *var/log/messages*, чтобы выявить источник

проблемы. Довольно трудно проверить, в точности ли соответствуют полномочия, которыми вы располагаете, необходимым. Ниже приводятся результаты рекурсивного выполнения команды ls на системе, где служба BIND запущена в окружении chroot:

```
root@gemini:/var/named/jail# ls -lR
.:
total 12
drwxr-xr-x 2 root root 4096 Sep 20 12:45 dev/
drwxr-xr-x 2 root root 4096 Sep 20 13:08 etc/
drwxr-xr-x 5 root root 4096 Sep 20 13:04 var/

./dev:
total 0
crw-rw-rw- 1 root root 1, 3 Jul 17 1994 null
crw-r--r-- 1 root root 1, 8 Dec 11 1995 random

./etc:
total 16
-rw-r--r-- 1 root root 1017 Sep 20 12:46 localtime
-rw-r--r-- 1 root root 1381 Sep 20 13:08 named.conf
-rw------- 1 root root 77 Sep 11 04:22 rndc.key

./var:
total 12
drwxr-xr-x 4 root root 4096 Sep 20 13:01 named/
drwxr-xr-x 2 named named 4096 Sep 20 13:15 run/

./var/named:
total 8
drwx------ 3 named named 4096 Sep 20 13:03 data/
drwxr-xr-x 2 named named 4096 Sep 20 12:43 slave/

./var/named/data:
total 42
-rw-r--r-- 1 root root 381 Apr 30 16:32 localhost.rev
-rw-r--r-- 1 root root 2769 Sep 20 13:03 named.ca
-r--r--r-- 1 root root 1412 Sep 17 16:44 nocat.net

./var/named/slave:
total 0

./var/run:
total 4
-rw-r--r-- 1 named named 6 Sep 20 13:15 named.pid
```

В окружении chroot демоны выполняются обособленно от остальной части системы, но даже в этом случае его нельзя назвать щитом ее безопасности. Можете использовать этот прием в качестве отправной точки, но не забудьте учесть все нюансы, связанные со сложностями запуска и использования службы BIND.

Дополнительная информация:

- *http://www.losurs.org/docs/howto/Chroot-BIND.html*;
- «DNS BIND», четвертое издание (O'Reilly).

СОВЕТ №77 Представления в BIND 9

(Изменение вида возвращаемых сервером DNS выходных данных в зависимости от источника запроса)

До недавних пор выдача одного представления зоны для одной группы узлов и другого представления для другой группы требовала наличия нескольких серверов доменных имен и запуска нескольких процессов на одном хосте с весьма сложными настройками.

В службе BIND 9 появилась новая функция под названием *представление*, позволяющая легко предоставлять доступ к различным версиям зоны и даже различным настройкам сервера доменных имен.

Базовый синтаксис

Способ настройки представлений в новой версии BIND 9 в выражении view.view требует задания имени представления в качестве аргумента и одного обязательного подвыражения, соответствующего клиентам. В качестве имени представления лучше всего использовать мнемоническое выражение, помогающее идентифицировать данное представление. Можно воспользоваться, например, именами internal, external или Internet. Если хотите воспользоваться именем, которое может конфликтовать с зарезервированным словом в *named.conf*, не забудьте взять его в кавычки.

Подвыражение с именами клиентов должно включать в себя список адресов в качестве аргументов. Только те клиенты, которые пришлют запрос с IP-адреса, числящегося в данном списке, увидят такое представление данных. Если IP-адрес клиента, обратившегося с запросом, входит в списки адресов для нескольких представлений, для него будет выдано то представление, соответствие для которого будет найдено первым.

Приведем пример простейшего представления:

```
view «global» {
match-clients {any; };
};
```

Это выражение для представления не несет никакой смысловой нагрузки, поскольку применяется ко всем запросам (при условии отсутствия других выражений) и не включает других подвыражений, помимо match-clients, которые бы отличали настройку этого представления от настроек по умолчанию.

Если не задано частное подвыражение в представлении, оно унаследует все глобальные настройки для этого подвыражения. Поэтому, например, если в представлении не выключена рекурсия, оно унаследует настройки рекурсии из параметров, заданных в подвыражении рекурсии, либо, если таковое отсутствует, примет настройки по умолчанию – со включенной рекурсией.

Рассмотрим пример, в котором рекурсия отключена:

```
view «external» {
match-clients { any; };
recursion no;
};
```

А так будет выглядеть полностью файл *named.conf*, содержащий только что рассмотренные нами выражения для представлений:

```
/*
* Сервер доменных имен BIND 9 допускает выполнение рекурсивных запросов
* с локального узла сети, запрещая их выполнение откуда-либо еще
*/

options {
directory «/var/named»;
};

view «localhost» {
match-clients { localhost; };
recursion yes; /*значение по умолчанию */
};

view «external» {
match-clients { any; };
recursion no;
};
```

Этот сервер доменных имен разрешает выполнение рекурсивных запросов только в том случае, если они поступили с локальной системы. Таблица контроля доступа на локальной системе [ACL] содержит IP-адреса узла, на котором запущен сервер доменных имен, а также адрес интерфейса обратной связи. Запросы со всех остальных адресов будут обслуживаться как нерекурсивные.

Ниже приводится файл с похожими настройками, отличающимися тем, что теперь рекурсивные запросы разрешены из локальной сети, но запрещены из сети интернет:

```
/*
* Один и тот же сервер доменных имен, обслуживающий и локальную сеть,
* и сеть интернет
*/
```

```
options {
directory «/var/named»;
};

view «internal» {
match-clients { localnets; };
recursion yes; /*значение по умолчанию */
};

view «external» {
match-clients { any; };
recursion no;
};
```

В данной конфигурации используется встроенный список контроля доступа localnets, который определяется заранее, как и все сети, к которым напрямую подключен наш сервер доменных имен.

Определение зон в представлениях

Чтобы представить версию зоны только для конкретного ряда запросов, необходимо определить зоны внутри представления, т.е. добавить выражения для зон также, как это сделано для подвыражений представления. Во всем остальном синтаксис подвыражения не будет отличаться от предыдущего примера, как будто имеем дело с выражением верхнего уровня. Учтите, что если будет определена хотя бы одна зона в представлении, ее придется определять во всех представлениях.

```
/*
* Сервер доменных имен, предоставляющий различные зональные данные
* для различных сетей
*/

options {
directory «/var/named»;
};

view «internal» {
match-clients { localnets; };
recursion yes; /*значение по умолчанию */

zone «oreilly.com» {
type master;
file «db.oreilly.com.internal»;
allow-transfer { any; }
};
};

view «external» {
match-clients { any; };
recursion no;
```

```
zone «oreilly.com» {
type master;
file «db.oreilly.com.external»;
allow-transfer { none; }
};
};
```

Обратите внимание: зона oreilly.com определена как в представлении internal, так и в представлении external, но при этом для зоны oreilly.com используются разные файлы данных — db.oreilly.com.internal для представления internal и db.oreilly.com.external для представления external. По-видимому, содержимое этих файлов также должно отличаться.

Если вы предпочитаете использовать различные поддиректории, чтобы хранить данные для внутренней и внешней зон, можете организовать и это. Например, подвыражением файла для oreilly.com в представлении internal может являться выражение file «db.oreilly.com.internal»; а для представления external — file «db.oreilly.com.external».

Точно так же можно хранить файлы данных в разных местах: для представления internal — в папке */var/named/internal*, а для представления external — в папке */var/named/external*.

Представления на ведомых серверах доменных имен

Единственное затруднение в создании представлений связано с настройкой ведомых серверов доменных имен. Большинство людей сначала настраивают главный ведущий сервер доменных имен с помощью нескольких представлений, а затем пытаются сконфигурировать ведущий сервер на использование тех же представлений. К сожалению, когда ведомый сервер пытается выполнить зональный переход или переход между двумя версиями одной зоны с главного ведущего сервера доменных имен, он увидит только ту версию зоны, которая определена в представлении, доступном для IP-адреса ведомого сервера.

В качестве решения этой проблемы можно предложить настройку псевдонима для IP-адреса ведомого сервера доменных имен, предоставляя в распоряжение сервера два IP-адреса, с которых возможна инициация зональных переходов. Затем необходимо настроить ведущий сервер для предоставления одного представления для первого IP-адреса ведомого сервера доменных имен, а второго представления для второго IP-адреса ведомого сервера доменных имен. После чего останется настроить ведомый сервер для инициации зональных переходов с соответствующего IP-адреса для каждого представления.

Приведем пример. Пусть ведомому серверу назначены два IP-адреса: 192.168.0.2 и 192.168.0.254. Ведущий сервер имеет один IP-адрес:

192.168.0.1. Рассмотрим для начала, как будет выглядеть файл *named.conf* для ведомого сервера:

```
options {
directory «/var/named»;
};

view «internal» {
match-clients { localnets; };
recursion yes;

zone «oreilly.com» {
type slave;
masters { 192.168.0.1; };
transfer-source 192.168.0.2;
file «internal/bak.oreilly.com»;
allow-transfer { any; }
};
};

view «external» {
match-clients { any; };
recursion no;

zone «oreilly.com» {
type slave;
masters { 192.168.0.1; };
transfer-source 192.168.0.254;
file « external/bak.oreilly.com «;
allow-transfer { none; }
};
};
```

Обратите внимание: ведомый сервер настроен на инициирование переходов на oreilly.com внутри представления internal с IP-адреса 192.168.0.2 и внутри представления external с IP-адреса 192.168.0.254.

Теперь рассмотрим файл *named.conf* для ведущего сервера доменных имен:

```
options {
directory «/var/named»;
};

view «internal» {
match-clients { 192.168.0.254; localnets; };
recursion yes; /*значение по умолчанию */

zone «oreilly.com» {
type master;
file «internal/db.oreilly.com»;
allow-transfer { any; }
};
```

```
};

view «external» {
match-clients { any; };
recursion no;

zone «oreilly.com» {
type master;
file «external/db.oreilly.com»;
allow-transfer { 192.168.0.254; }
};
};
```

Обратите внимание: подвыражение match-clients в представлении internal однозначно определяет адрес 192.168.0.254 в представлении external, вычеркивая его из списка адресов соответствия. Каждый раз, когда ведомый сервер будет инициировать зональные переходы с этого IP-адреса, он будет получать доступ к версии сайта oreilly.com, описанной в файле данных зоны */var/named/external/db.oreilly.com*.

СОВЕТ №78 Настройка кэширования DNS с правами для локальных доменов

(Заставьте службу BIND эффективно работать на сервере, выполняющем кэширование и перенаправление)

Запуск службы BIND может оказаться весьма непростой задачей, если сеть имеет достаточно сложную топологию. Несколько демилитаризованных зон, открытые и локальные IP-адреса и делегированные домены способны вылиться для администратора DNS в полноценный рабочий день, если организация имеет достаточно крупный сайт. При необходимости упростить структуру сети см. раздел *«Представления в BIND 9»* **[Совет № 77]**, а если чувствуете в себе дух приключений, попробуйте задать маски для соответствия доменам и делегации, как описано в разделе *«Обслуживание узлов: массовый веб-хостинг при помощи подстановочных символов, прокси-серверов и перенаправления»* **[Совет № 100]**.

Но в большинстве средних и небольших инсталляций служба BIND на самом деле необходима только в двух целях: чтобы действовать в качестве официального источника информации для домена и обеспечить перенаправление всех остальных запросов на другой DNS-сервер.

Ниже приводится простая, но завершенная версия файла *named.conf*, выполняющего вышеупомянутые задачи:

```
options {
directory «/var/named»;
pid-file «/var/run/named.pid»;
statistics-file «/var/named/named.stats»;
};
```

```
logging {
channel default_out {
file «/var/log/named.log»;
};

category default { default_out; };
category config { default_out; };
category xfer-in { default_out; };
category xfer-out { default_out; };
category lame-servers { null; };
};

zone «0.0.127.in-addr.arpa» in {
type master;
file «data/localhost.rev»;
};

zone «.» {
type hint;
file «rootservers.cache»;
};

// Список надежных доменов должен быть задан здесь

zone «nocat.net» {
type master;
file «data/nocat.net»;
};
```

Таким образом, теперь мы будем считаться надежным узлом для домена nocat.net, данные для которого хранятся в файле */var/named/data/nocat.net*. Запросы к домену, отличному от nocat.net, или к интерфейсу обратной связи 127.0.0.0, будут автоматически перенаправляться в соответствии со списком главных серверов, перечисленных в файле *rootservers.cache*. Подходящий кэш главного сервера должен находиться рядом со службой BIND, но если не сможете найти его, попробуйте следующую команду:

```
# dig @a.root-servers.net > rootservers.cache
```

Если в локальной сети отсутствует неограниченный доступ к сети интернет (например, в ней имеется ограничивающий брандмауэр) перенаправление запросов на надежные серверы может не сработать. В таком случае, попробуйте задать явное правило перенаправления вместо зоны типа hint, описанной в предыдущем примере:

```
zone «.»;
type forward;
forward only;
forwarders { 192.168.1.1 };
}
```

Разумеется, вместо адреса 192.168.1.1 нужно будет подставить реальный IP-адрес сервера вашей сети. Таким образом, весь трафик DNS будет перенаправлен на ваш сетевой сервер DNS, который, как мы предполагаем, должен обладать правами на поиск доменов через брандмауэр.

Дополнительная информация:

- *«Представления в BIND 9»* **[Совет № 77]**;
- *«Обслуживание узлов: массовый веб-хостинг при помощи подстановочных символов, прокси-серверов и перенаправления»* **[Совет № 100]**;
- «DNS и BIND», 4-е издание (O'Reilly).

СОВЕТ №79

Распределение серверной загрузки при помощи циклического DNS

(Одновременное перенаправление трафика на несколько серверов при помощи циклического DNS)

Если вы обслуживаете чрезвычайно популярный сайт, то рано или поздно дойдете до предела, после которого ваш сервер просто откажется обслуживать другие запросы. На хакерском жаргоне это называют *slashdot-эффектом* (назван так в честь популярной службы новостей Slashdot – *http://slashdot.org/*, упоминание веб-ссылки на которую приводит к резкому росту посещаемости вышеупомянутого сайта, часто делая его недоступным — *прим. перев.*). Добавление ОЗУ, обновление процессора, использование более быстрых дисковых накопителей и шин способно помочь в краткосрочной перспективе, но рано или поздно можно столкнуться с ситуацией, когда один компьютер просто окажется не в состоянии справиться с необходимым объемом работ.

Единственный способ обойти подобные ограничения заключается в распределении нагрузки между несколькими компьютерами. Добавление второго (или третьего) сервера к доступному пулу компьютеров позволит повысить не только быстродействие, но и стабильность работы сети. Наличие второй (или третьей) резервной постоянно работающей системы позволяет в случае возникновения проблем распределить работу между оставшимися серверами без простоя.

Вероятно, самый удобный способ распределения общественного трафика сводится к тому, чтобы заставить это самое общество самому заняться решением задачи по распределению трафика. Благодаря свойствам циклического DNS входящие запросы, предназначенные для одного узла сети, могут быть переданы для обслуживания произвольному количеству IP-адресов.

В службе BIND 9 сделать это не сложнее, чем добавить несколько записей A для отдельного узла сети. Предположим, что используем это свойство для файла зоны oreillynet.com:

```
www 60 IN A 208.201.239.36
www 60 IN A 208.201.239.37
```

Теперь, когда на узле сети осуществляется поиск строки www.oreillynet.com среди записей DNS, в половине случае мы увидим на экране следующие сообщения:

```
rob@caligula:~$ host www.oreillynet.com
www.oreillynet.com has address 208.201.239.36
www.oreillynet.com has address 208.201.239.37
```

во второй половине случаев порядок обнаружения записей будет такой:

```
rob@caligula:~$ host www.oreillynet.com
www.oreillynet.com has address 208.201.239.37
www.oreillynet.com has address 208.201.239.36
```

Поскольку большинство приложений обычно используют только первый найденный на сервере DNS адрес, подобное решение весьма действенно. Каждый сервер обслуживает приблизительно половину запросов, то есть нагрузка на каждый сервер снижается вдвое. Мы ограничим предписанное время жизни пакета (TTL) 60 секундами, чтобы предотвратить вмешательство любых кэширующих DNS-серверов, благодаря чему количество запросов, поступающих на каждый сервер, будет более или менее одинаковым.

Распределение нагрузки имеет смысл только в том случае, если данные на всех серверах синхронизированы. В противном случае, при попадании на один сервер веб-браузер будет получать одну версию веб-страницы, а при попадании на второй – совершенно другую. Чтобы избежать этого, прочтите раздел *«Синхронизация отдельных частей файловой системы при помощи утилиты rsync»* **[Совет № 41]**.

Учтите также, что реальное взятие под контроль одного IP-адреса в случае, если один из узлов сети не в состоянии выполнять свои обязанности, – задача непростая. Если один из серверов перестанет выполнять свои функции, то нельзя будет изменить запись DNS и придется ожидать, пока эта новость распространится по интернету. Вам же необходимо, чтобы кто-то взял на себя выполнение задач вышедшего из строя сервера немедленно. Один из методов реализации данной задачи рассмотрен в разделе *«Контроль над IP-адресами»* **[Совет № 63]**.

Дополнительная информация:

- *«Синхронизация отдельных частей файловой системы при помощи утилиты rsync»* **[Совет № 41]**;
- *«Контроль над IP-адресами»* **[Совет № 63]**.

СОВЕТ №80

Запуск собственного домена верхнего уровня

(Настройте собственный домен верхнего уровня в службе BIND для упрощения навигации)

Если вы работаете администратором сети, в которой применяется частная адресация, то можете в конце концов заработать раздвоение личности, пытаясь обслужить файлы зон для корректного разделения внутренних и внешних IP-адресов. Благодаря появлению представлений (см. раздел *«Представления в BIND 9»* **[Совет № 77]**) поддержка нескольких диапазонов адресов в одном домене значительно упростилась.

Но советуем также испытать на практике легкость настройки собственного домена верхнего уровня. Обычно записи, касающиеся зон, в файле *named.conf* выглядят следующим образом:

```
zone «oreillynet.com» {
type master;
file «data/oreillynet.com»;
};
```

Эта запись подходит для официального DNS-сервера домена oreillynet.com. Вообще же, количество реальных доменов верхнего уровня (например, .com, .net, .org, .int и т.д.), известных как основные DNS-серверы, ограничено мистическим числом 13. Несмотря на то что создание собственного домена верхнего уровня приведет к недоступности ваших серверов из сети интернет, это может оказаться весьма удобным для работы внутри локальной сети.

Предположим, есть группа компьютеров, составляющая локальную сеть 192.168.1.0/24. Получить доступ к ним непосредственно из сети интернет невозможно, да никто и не стремится афишировать информацию DNS, которая может заинтересовать потенциальных сетевых взломщиков. Попробуйте воспользоваться нестандартными доменами верхнего уровня:

```
zone «bp» {
type master;
file «data/bp»;
allow-transfer { 192.168.1/24; };
allow-query { 192.168.1/24; };
};
```

где bp, в данном случае, сокращение от backplane — задняя панель, но чтобы не отвлекаться от темы обсуждения, считайте, что это просто еще одна разновидность сокращений. Добавив этот фрагмент в свой файл описания с зонами, настройте мастер-запись для bp, как для любого другого домена:

```
$TTL 86400
@ IN SOA ns.bp. root.homer.bp. (
2002090100 ; Serial
10800 ; Обновлять через 3 часа
3600 ; Повторная попытка через 1 час
604800 ; Срок действия (1 неделя)
60 ; Отрицательное время окончания срока действия
)

IN NS ns.bp.

ns IN A 192.168.1.1

homer IN A 192.168.1.10
bart IN A 192.168.1.11
lisa IN A 192.168.1.12
```

Перезагрузите службу named. Теперь сможете выполнить команду ping homer.bp. Если хотите использовать другие серверы доменных имен для обслуживания ведомых копий своего домена верхнего уровня, добавьте их стандартным способом:

```
zone «bp» {
type slave;
file «db.bp»;
masters { 192.168.1.1; };
};
```

Таким образом, можете внедрить использование своего нового домена верхнего уровня по всей архитектуре локальной сети. Если используются туннели через интернет (как в разделе *«Туннелирование: инкапсуляция IPIP»* **[Совет № 50]**) для получения доступа к удаленному офису или компьютерам друзей, то теоретически можете неограниченно расширять поддержку своего домена верхнего уровня.

СОВЕТ №81

Мониторинг состояния MySQL при помощи утилиты mtop

(Отображение потоков MySQL в режиме реального времени в формате, подобном утилите top)

Во многом похожая на свой аналог top, утилита mtop позволяет наблюдать в окне терминала статистику по серверу MySQL в режиме ре-

ального времени. На загруженном сервере базы данных невозможно просмотреть подробности например то, какие запросы выполняются в текущий момент времени и съедают все ресурсы.

Для запуска утилиты mtop задайте следующие переключатели в командной строке:

```
mysql --dbuser=monitor --password=notelling
```

Естественно, вместо dbuser и password следует подставить имя пользователя БД и пароль. Если mtop запускается с другого узла сети, не того, на котором расположен сервер базы данных, необходимо добавить переключатель --host={mysql_host}. После запуска утилиты на экран выводится информация, напоминающая сведения, возвращаемые утилитой top, которая будет обновляться каждые несколько секунд:

```
load average: 0.72, 0.47, 0.26 mysqld 3.23.51 up 33 day(s), 4:48 hrs
2 threads: 2 running, 0 cached. Queries/slow: 71.5K/0 Cache Hit 99.99%
Opened tables: 42 RRN: 4.0M TLW: 0 SFJ: 0 SMP: 0

ID USER HOST DB TIME COMMAND STATE INFO
26049 root localhost test Query shoe full processlist
26412 root localhost nocat 1 Query Writing to n select * from Member where User like
'%rob%'
---
```

Отсюда можно узнать: какие пользователи подключены в текущий момент времени; узлы сети, с которых было установлено подключение и к какой базе данных, а также как долго выполнялся каждый поток (в секундах). Число слева представляет собой идентификатор потока для каждого активного запроса, но не идентификатор процесса mysql. Если некоторый запрос рискует попасть в категорию медленно выполняющихся, строка выделяется фиолетовым цветом, а если это все-таки произошло, цвет строки изменится на желтый. Если поток продолжает оставаться активным после двойного превышения значения параметра MySQL long_query_time, строка будет выделена красным цветом. Благодаря этому можно с первого взгляда определить, какие запросы выполняются слишком долго.

Как и для утилиты top, в mtop можете вызвать справку по доступным сочетаниям клавиш, нажав во время ее работы клавишу ?:

```
mtop ver 0.6.2/2000905, Copyright © 2002, Marc Prewitt/Chelsea Networks

A top users display for mysql

These single-character commands are available:
/* Доступные односимвольные команды:
```

```
q – quit
/* выход
? – help; show this text
/* вызов помощи; то, что сейчас видите на экране
f – flush status
/* очистка статуса
k – kill processes; send a kill to a list of ids
/* уничтожение процессов; применяет команду kill к списку идентификаторов
s – change the number of seconds to delay between updates
/* изменяет период обновления
m – toggle manual refresh mode on/off
/* включает/отключает режим обновления вручную
d – filter display with regular expression (user/host/db/command/state/info)
/* фильтрует выводимую информацию согласно регулярным выражениям
h – display process for only one host
/* отображает процессы только для одного хоста
u – display process for only one user
/* отображает процессы только для одного пользователя
i – toggle all/non-Sleeping process display
/* переключатель отображения всех / только активных процессов
o – reverse the sort order
/* обратный порядок сортировки
e – explain a process; show query optimizer info
/* вывести детали работы процесса; показать информацию оптимизатора запро-
сов
t – show mysqld stats (show status/mysqladmin ext)
/* показать статистику по mysqld
v – show mysqld variables (show variables/mysqladmin vars)
/* показать переменные mysqld
z – zoom in on a process, show sql statement detail
/* выделить процесс и показать детали выражения sql
r – show replication status for master/slaves
/* отобразить статус репликации для ведущего/ведомого
```

Вероятно, чаще всего используются функции explain (e) и kill (k). При нажатии клавиши e на запрос об ID потока наберите номер интересующего запроса — и увидите, какие операции в реальности выполняет сервер mysql при его обслуживании:

```
Id: 27134 User: root Host: localhost Db: nocat Time: 0
Command: Query State: cleaning up

select *
From Member
WHERE User like '%rob%'

table |type |possible_keys |key | ken_len| ref | rows|
Member |ALL | | | | | 9652|where used
```

Аналогично, ключ k позволяет уничтожать поток путем задания его ID. Это чрезвычайно удобно для уничтожения долго выполняющихся или требовательных и плохо оптимизированных запросов,

без необходимости многократно запускать mysqladmin из командной строки.

Один из методов, позволяющий избежать ввода имени пользователя и пароля в командной строке, заключается в создании пользователя mysqltop с ограниченными привилегиями. Из документации по Perl (perldoc) mtop:

> Самый удобный способ настройки системы — воспользоваться утилитой mtop для создания пользователя базы данных с именем mysqltop без пароля. В целях безопасности, этот пользователь должен быть лишен всех привилегий, кроме Process_priv, значение для которой необходимо установить в Y.

Для предоставления пользователю mysqltop подобных привилегий необходимо запустить на выполнение следующий программный код из приглашения командной строки MySQL:

```
mysql> grant select on test.* to mysqltop;
mysql> grant select on test.* to mysqltop@localhost;
mysql> update user set process_priv='y' where user='mysqltop';
mysql> flush privileges;
```

Установив утилиту mtop и прописав путь к ней в переменной PATH, вы сбережете свое время при помощи одной только программы mysqladmin.

Дополнительная информация:

- пакет mtop, который можно загрузить из сети интернет по адресу *http://mtop.sourceforge.net/* (для его работы требуется установить Perl 5);
- curses.pm доступен в полной сети архивов по Perl – CPAN.

СОВЕТ №82

Настройка репликации в MySQL

(Работа с актуальными копиями своей базы данных для повышения быстродействия и обеспечения избыточности)

С выходом версии 3.23.33 в MySQL была реализована репликация баз данных. Репликация организовывалась при помощи двоичного журнала операций над базой данных на одном компьютере (ведущем), с которым выполняли синхронизацию другие компьютеры (ведомые). MySQL версии 3.23 позволял осуществлять только однонаправленную репликацию, т. е. вносимые изменения на ведущем сервере копировались на все ведомые серверы; однако изменения, внесенные на ведомых серверах, не могли быть сохранены на ведущем. При возникновении необходимости в выполнении двунаправ-

ленной репликации обратите внимание на прогрессивный программный код MySQL 4.

Чаще всего репликация баз данных используется для распределения нагрузки по обслуживанию запросов к БД между несколькими компьютерами. Это помогает не только повысить производительность, но и добавляет избыточность на случай, если один из серверов базы данных выйдет из строя. Используемое вами приложение должно быть достаточно умным, чтобы понимать, что запросы на запись должны обслуживаться исключительно ведущим сервером, иначе между реплицированными копиями могут возникнуть несоответствия. Это не лучший способ времяпрепровождения, поэтому проверьте весь ваш программный код, чтобы запись данных из него велась *только* в главную базу.

Ниже приводятся девять простых действий, необходимых для того, чтобы заставить работать репликацию на MySQL версии 3.23.33 (либо более поздней). Но сначала убедитесь, что на всех ваших компьютерах запущена одна и та же версия MySQL, желательно, самая свежая и стабильная.

1. Решите, какой из серверов базы данных будет ведущим, а какие — ведомыми. Обычно в качестве ведущего сервера базы данных выбирают наиболее мощный компьютер. Если вы владеете чрезвычайно популярным сайтом, который должен обслуживать большое количество запросов к базе данных, настоятельно рекомендуем приобрести многопроцессорную систему с большим количеством ОЗУ и аппаратным RAID-массивом. Иначе от нехватки ресурсов пострадают все динамические приложения, поскольку, независимо от количества компьютеров, на которые предполагается распределить нагрузку, они окажутся заблокированными до того момента, пока в базе данных не будут сохранены изменения.
2. Создайте отдельного пользователя на ведущем сервере для выполнения репликации базы данных. Он ничем не будет отличаться от других пользователей MySQL, за исключением того, что будет обладать полномочиями FILE на доступ ко всем базам данных, которые подвергнутся репликации:

```
mysql> grant FILE on webdb.* to replicant@'%.mynetwork.edu' identified
by 'seCret';
```

3. Включите запись двоичного файла журнала на ведущем сервере и выберите, какие базы данных должны быть реплицированы. Добавьте в раздел [mysql] файла */etc/my.cnf*, размещенного на ведущем сервере, следующие строки:

```
log-bin
server-id=1
```

Кроме этого, включите строку binlog-do-db на всех базах данных, которые должны быть доступны в результате репликации:

```
binlog-do-db=webdb
```

Теперь отключите MySQL и немедленно запустите его вновь, чтобы изменения вступили в силу:

```
root@db:/usr/local/mysql# mysqladmin shutdown; ./bin/safe_mysqld&
```

4. Завершите работу с базой данных на главном сервере и выполните архивирование директории */data*. Для этого недостаточно воспользоваться последними дампами MySQL (mysqldumps); необходимо создать реальную копию директории */data*. Весь процесс может занять от нескольких секунд до нескольких минут, в зависимости от объема информации, хранимой в базах данных:

```
root@db:/usr/local/mysql# mysqladmin shutdown; tar cvf ~/data.tar data/
```

Сервер базы данных будет недоступен в течение всего периода копирования, поэтому выполнять подобные действия следует в часы минимальной загрузки.

5. Заново включите доступ к базе данных на главном сервере. Если выполнение команды tar завершилось успешно, запустите службу MySQL:

```
root@db:/usr/local/mysql# ./bin/safe_mysqld&
```

6. Скопируйте архив на каждый из ведомых серверов, убедившись в том, что служба MySQL на них не запущена. Для этого запустите на каждом ведомом сервере следующую команду:

```
root@slave:~# mysqladmin shutdown; scp db:data.tar .
```

7. Присвойте каждому ведомому серверу уникальный идентификатор и разрешите выполнение репликации. С этой целью понадобится включить следующие строки в раздел [mysql] файла */etc/my.cnf* на каждом ведомом сервере:

```
master-host=db.mynetwork.edu
master-user=replicant
master-password=seCret
server-id=10
```

Параметр server-id должен являться уникальным для каждого ведомого сервера. В качестве идентификатора можете выбрать любое целое число, которое ранее не использовалось для идентификации ведомого или ведущего сервера.

Если какой-либо сервер выступает исключительно в роли ведомого по отношению к главному и не обслуживает никаких запросов на чтение/запись собственных баз данных, можно несколько повысить производительность, включив следующие опции:

```
low-priority-updates
scip-bdb
delay-key-write-for-all-tables
```

Таким образом нивелируется влияние накладных расходов, связанных с выполнением некоторых фрагментов программного кода, используемых только при записи данных в базу. А т.к. данный сервер является ведомым и вести запись в базу данных ему не понадобится, службе MySQL не нужно находиться в постоянной готовности к обслуживанию запросов на запись.

8. Распакуйте архив с данными на каждом ведомом сервере:

```
root@slave:/usr/local/mysql# mv data data.old; tar vxf ~/data.tar
```

Убедитесь, что новой директорией *data/* и всем ее содержимым владеет пользователь и группа mysql:

```
root@slave:/usr/local/mysql# chown -R mysql.mysql data/
```

Если вам не нужно ничего из оригинальной папки *data/* на ведомых серверах, можете ее удалить с помощью команды rm -rf (вместо того чтобы архивировать ее в *data.old/*).

9. Запустите MySQL на каждом ведомом сервере, затем просмотрите журнал ошибок на случай возможных неполадок. После этого можете запускать MySQL на постоянную работу:

```
root@slave:/usr/local/mysql# ./bin/safe_mysqld&
```

лишь иногда просматривая журналы ошибок:

```
root@slave:/usr/local/mysql# tail -f data/slave.err
```

Естественно, вместо slave следует подставить реальное имя компьютера, выступающего в роли ведомого сервера. Если вы увидите сообщение «Репликация начата с позиции XYZ» (Starting replication at position XYZ) — значит, репликация началась! Теперь попробуйте сохранить изменения в главной базе данных, а затем отправить запрос к ведомым серверам, чтобы проверить, отразились ли изменения на них. Обновление реплицированных баз данных должно происходить практически мгновенно. В случае возникновения каких-либо проблем с репликацией, отчет об ошибках можно прочесть в журнале mysql того ведомого сервера, на котором возникли проблемы.

Обычно проблемы с репликацией вызываются несоответствием полномочий (проверьте синтаксис команды GRANT на втором шаге) или данные перестают быть синхронизированы на ведомых серверах по какой-то другой причине — в таком случае повторите действия 4, 5, 6, 8 и 9 *очень внимательно*. Не спешите и всегда проверяйте журналы ошибок mysql на каждом из ведомых серверов.

Если вы собираетесь в будущем подключать дополнительные ведомые серверы, тогда храните под рукой копию *data.tar*. Можете воспользоваться ею в любое время для создания нового ведомого сервера (после чего он будет синхронизирован с базой данных на главном сервере по сети при помощи двоичного файла журнала операций, выполненных над базой данных с момента создания архива *data.tar*.)

Включить репликацию и поддерживать базу данных со всеми реплицированными копиями в работоспособном состоянии – совершенно разные понятия. Если приложение сохраняет изменения только в главной базе данных, а аппаратное обеспечение достаточно надежно – проблем возникать не должно. Но стоит копии базы данных на одном из ведомых серверов потерять синхронизацию по какой-то причине, понадобится обратиться к дополнительным источникам информации помимо данного параграфа. Рекомендуем вначале прочесть дополнительную литературу, прежде чем перейти к решению более сложных ситуаций.

Дополнительная информация:

- «Справочное руководство по MySQL» (O'Reilly);
- онлайновая документация по MySQL, размещенная по адресу *http://www.mysql.com/doc/en/Replication.html*.

СОВЕТ №83 Восстановление отдельной таблицы из большого дампа MySQL

(Метод восстановления отдельных таблиц mysql из дампа MySQL)

Как любой хороший администратор, вы каждый день честно сохраняете дамп всех таблиц mysql в сжатом виде (вероятно, для последующей распаковки сценарием восстановления из резервных копий). Скорее всего, в планировщике главного сервера базы данных или на одном из ведомых серверов содержится следующая запись:

```
for x in 'mysql -Bse show databases'; do
mysqldamp $x | gzip -9 > /var/spool/mysqldump/$x.'date +%Y%m%d'.dz
done
```

Это спасет вас на случай катастрофы с реальной базой данных. Однако, если размер базы данных слишком возрастет, процедура восста-

новления из разбитой на отдельные части резервной копии может оказаться достаточно сложной. Для базы данных, содержащей несколько миллионов записей, дампы могут превратиться в кучи информации, которые необходимо как-то обрабатывать. Как можно найти и восстановить одну таблицу среди нескольких сотен мегабайт архивированного дампа?

С этой целью можно воспользоваться простым методом: напишите сценарий на языке Perl под названием extract-table, который будет содержать следующие строки:

```
#!/usr/bin/perl -wn
BEGIN { $table = shift @ARGV }
print if /^create table $table\b/io .. /^create table (?!$table)\b/io;
```

Чтобы извлечь таблицу Users из дампа базы данных под названием randomdb, воспользуйтесь следующим фрагментом программного кода:

```
# zcat /var/spool/mysqldump/randomdb.20020901.gz | extract-table Users > ~/
Users.dump
```

Теперь можете упростить процесс восстановления таблицы Users еще больше:

```
# mysql randomdb -e «drop table Users»
# mysql randomdb < ~/Users.dump
```

СОВЕТ №84

Настройка сервера MySQL

(Практические шаги, позволяющие добиться максимальной производительности от сервера MySQL)

Многие администраторы Linux внезапно оказываются в роли «администратора БД, проживающего по месту службы», когда больше никто не хочет или не может взять на себя эти обязанности. Часто сотрудники, специализирующиеся на настройке и обслуживании баз данных в работоспособном состоянии, вообще не касаются сферы ответственности системных администраторов, но системного администратора Linux могут привлекать к обслуживанию баз данных с минимальным обучением (и, очевидно, без прибавки жалованья). Хотя изучение данного параграфа не превратит вас в эксперта по администрированию баз данных, но научит некоторым практическим приемам, которые помогут повысить их производительность в реальных условиях.

Далее рассматриваются пять действий, которые необходимо предпринять для оптимизации работы сервера MySQL, перечисленные в порядке увеличения сложности и эффективности.

1. Запустите команду mysqlcheck -o database. Она позволяет выполнить оптимизацию таблиц, уменьшить потерянное пространство путем «дефрагментации» информации, хранимой в БД. Такую команду полезно запускать, если недавно менялась структура базы данных либо из нее удалялись большие объемы информации.
2. Измените приоритет выполнения демона mysqld. Если в вашем распоряжении имеется выделенный компьютер для работы сервера MySQL, можете поручить планировщику запускать mysqld с повышенным приоритетом. В оперативном руководстве по серверу mysql рекомендуется добавить с этой целью в сценарий safe_mysqld команду:

```
renice -20 $$
```

Полезно также добавить следующий фрагмент кода:

```
NOHUP_NICENESS=«nohup»
if test -w /
then
NOHUP_NICENESS='nohup nice 2>&1'
if test $? -eq 0 && test x«$NOHUP_NICENESS» != x0 && nice --1 echo foo > /
dev/null 2>&1
then
NOHUP_NICENESS=«nice --$NOHUP_NICENESS nohup»
else
NOHUP_NICENESS=«nohup»
fi
fi
```

а затем заменить его на простую строку:

```
NOHUP_NICENESS=«nohup nice --20»
```

Теперь сценарий safe_mysqld и все остальные процессы mysqld будут запускаться с наивысшим приоритетом. Естественно, это становится возможным только за счет отъема процессорного времени у других программ, поэтому, если вам понадобится запускать на системе с mysqld другие процессы, выберите в качестве приоритета большее значение (где-то в промежутке от -10 до -5).

3. Создайте индексы. Если приходится работать с долго выполняющимися запросами, один из способов оптимизации сводится к добавлению подходящих индексов. Столкнувшись с долго выполняющимся запросом (см. раздел *«Мониторинг состояния MySQL с помощью утилиты mtop»* **[Совет № 81]**), подумайте о создании релевантного индекса:

```
mysql> create index name on Member (Name(10));
```

Индексирование баз данных всегда представляет собой компромисс между занимаемым объемом дискового пространства и производительностью, но в наш век невысокой стоимости дисковых носителей нет никакого смысла пренебрегать индексами. Как правило, наличие излишнего количества индексов не повредит, а вот их отсутствие при выполнении линейного поиска (либо уникального вхождения) в большой таблице способно существенно перегрузить сервер.

4. Проверьте переменные на сервере. Значения переменных, используемых по умолчанию, предназначены для обеспечения безопасных и разумных настроек для компьютеров с не самым новым аппаратным обеспечением. Если имеется в наличии большое количество ОЗУ (512 Мб и больше), можно получить значительный выигрыш в производительности путем увеличения размеров буферов и кэша, заданных по умолчанию.

   Приведем список переменных, используемых нами на рабочем сервере базы данных (двухпроцессорная система на базе Pentium 4/1.0 ГГц с 2 Гб ОЗУ и большими быстродействующими дисками). Включите их в раздел [mysqld] файла */etc/my.cnf*:

```
set-variable = key_buffer=384M
set-variable = max_allowed_packet=1M
set-variable = table_cache=512
set-variable = sort_buffer=2M
set-variable = record_buffer=2M
set-variable = myisam_sort_buffer_size=64M
set-variable = tmp_table_size=8M

set-variable = max_connections=768
```

   Обратите внимание: многие из указанных параметров были взяты из кода примера *my-huge.cnf*, входящего в состав дистрибутива mysql (мы не стали их менять, так как они отлично работали в нашем случае). Поскольку на наших серверах запущен интерфейс базы данных Apache::DBI в переменной wait-timeout дополнительно задано значение в несколько минут:

```
set-variable = wait-timeout=120
```

   Таким образом мы оградим себя от возможного «зависания» потоков DBI и захвата всех доступных подключений max_connections.

5. Установите пакеты исправлений и обновлений для glibc. Когда вам станет недоставать возможностей стандартной установки glibc, подумайте о ее обновлении (см. раздел *«Оптимизация glibc, linuxthreads и ядра системы для сервера MySQL»* **[Совет № 86]**). Это позволит создавать более мелкие потоки и открывать большее

количество файлов. Хотя обычно это не представляет проблем, за исключением очень крупных и очень загруженных инсталляций MySQL.

Как и в случае с большинством других тем данной книги, тема настройки и администрирования баз данных слишком сложна, чтобы все связанные с ней вопросы можно было изложить на нескольких страницах. Более подробное обсуждение предмета можно найти в перечисленных ниже источниках информации.

Дополнительная информация:

- «Справочное руководство по MySQL» (O'Reilly);
- книга «MySQL» (New Riders);
- *http://www.mysql.com/doc/en/Linux.html*;
- *http://www.mysql.com/doc/en/SHOW_VARIABLES.html.*

СОВЕТ №85

Использование proftpd с источником аутентификации mysql

(Избавьте себя с помощью proftpd и mysql от необходимости создавать пользовательские бюджеты ftp)

Демон ftp под названием proftpd представляет собой мощную службу ftp, обладающую синтаксисом настройки, во многом напоминающим Apache. В нем доступны многие опции, отсутствующие в большинстве остальных демонов ftp, включая коэффициенты, виртуальный хостинг и модульную структуру, позволяющую создавать собственные модули.

Одним из таких дополнительных модулей является mod_sql, позволяющий службе proftpd использовать базу данных SQL в качестве серверного источника аутентификации. В настоящее время модуль mod_sql поддерживает MySQL и PostgreSQL. Это хороший способ заблокировать доступ к серверу, поскольку входящим пользователям придется проходить аутентификацию при подключении к самой базе данных, т.е. им не нужна учетная запись, позволяющая получать доступ к оболочке командной строки. В этом разделе мы изучим аутентификацию в службе proftpd, используемую для подключения к базе данных MySQL.

Загрузите исходные коды утилит proftpd и модуля mod_sql и выполните их компиляцию:

```
~$ bzcat proftpd-1.2.6.tar.bz2 |tar xf -
~/proftpd-1.2.6/contrib$ tar zvx ../../mod_sql-4.08.tar.gz
```

```
~/proftpd-1.2.6/contrib$ cd ..
~/proftpd-1.2.6$ ./configure --with-modules=mod_sql:mod_sql_mysql \
--with-includes=/usr/local/mysql/include/ \
--with-libraries=/usr/local/mysql/lib/
```

Разумеется, нужно будет указать реальный путь к дистрибутиву MySQL, если он отличается от указанного выше */usr/local/mysql/*. Откомпилируйте программный код и установите его:

```
rob@catlin:~/proftpd-1.2.6$ make && sudo make install
```

Создайте базу данных, которая будет использоваться proftpd (предполагается, что сервер MySQL уже настроен и работает),

```
$ mysqladmin create proftpd
```

и задайте к ней доступ только для чтения из proftpd:

```
$ mysql -e «grant select on proftpd.* to proftpd@localhost \
identified by 'secret';»
```

Создайте в базе данных две таблицы, следуя такой схеме:

```
CREATE TABLE users (
userid varchar(30) NOT NULL default '',
password varchar(30) NOT NULL default '',
uid int(11) default NULL,
gid int(11) default NULL,
homedir varchar(255) default NULL,
shell varchar(255) default NULL,
UNIQUE KEY uid (uid),
UNIQUE KEY userid (userid)
) TYPE=MyISAM
CREATE TABLE groups (
groupname varchar(30) NOT NULL default '',
gid int(11) NOT NULL default '0',
members varchar(255) default NULL
) TYPE=MyISAM
```

Впоследствии вы сможете быстро создавать таблицы, сохранив вышеприведенный программный код в файле под именем *proftpd.schema* и запуская команду mysql proftpd < proftpd.schema.

Сообщите службе proftpd о необходимости использовать эту БД для идентификации пользователей, дописав в файл */usr/local/etc/ proftpd.conf* следующие строки:

```
SQLConnectInfo proftpd proftpd secret
SQLAuthTypes crypt backend
SQLMinUserGID 111
SQLMinUserUID 111
```

Строка SQLConnectInfo вызывает форму БД для ввода пароля пользователя. Можно задать БД, размещенную на другом узле сети — и даже использующую другой порт — при помощи следующей команды:

```
SQLConnectInfo proftpd@хост_с_БД:5678 имя_пользователя пароль
```

Строка SQLAuthTypes позволяет создавать пользователей, пароли которых будут храниться в зашифрованном виде в стандартном формате шифрования Unix либо при помощи функции PASSWORD() сервера mysql. Учтите, что при использовании функции ведения протокола модуля mod_sql пароль может фигурировать в виде простого текста, поэтому доступ к журналам должен быть ограничен.

Строка SQLAuthTypes, в том виде, в котором она представлена здесь, не позволяет применять пустые пароли; если это нужно, включите в перечисление пустое ключевое слово.

Строки SQLMinUserGID и SQLMinUserUID определяют минимальный допустимый идентификатор пользователя или группы, разрешенный proftpd при входе в систему. Желательно задавать в нем значение, большее нуля, чтобы запретить вход в систему с правами пользователя root, но в то же время достаточно малое, чтобы иметь надлежащие права на доступ к файловой системе. В данном случае созданы пользователь и группа под названием www, uid и gid, для которых идентификаторы установлены в 111. Поскольку мы хотим, чтобы все разработчики могли входить в систему с аналогичными полномочиями, задано минимальное значение — 111.

Теперь мы готовы приступить к созданию пользователей в базе данных. Следующий фрагмент кода создаст пользователя jimbo с правами пользователя/группы www/www, направляя его после входа в систему в папку */usr/local/apache/htdocs/*:

```
mysql -e «insert into users values ('jimbo',PASSWORD('sHHH'),'111', \
'111','/usr/local/apache/htdocs','/bin/bash');» proftpd
```

Пароль пользователя jimbo перед сохранением будет зашифрован при помощи функции PASSWORD() сервера mysql. Строка /bin/bash передается в службу proftpd для дальнейшей передачи директивы RequireValidShell. Эта строка не имеет никакого отношения к предоставлению реального доступа к оболочке командной строки для пользователя jimbo.

На данном этапе уже можно запустить службу proftpd и войти в систему под учетной записью пользователя jimbo, указав в качестве пароля sHHH. Если во время установки подключения возникнут проблемы, попробуйте запустить proftpd в режиме отладки:

```
# proftpd -n -d 5
```

Следите за сообщениями, появляющимися при попытке установки подключения, чтобы определить источник проблемы. Проблемы почти всегда вызываются неправильными настройками в proftpd.conf, чаще всего связанными с установкой разрешений.

Модуль mod_sql способен на большее, помимо описанного в данном разделе. Он может подключаться к существующим базам данных mysql с произвольными именами таблиц, записывать в журнал все операции над базой данных, изменять условия поиска для пользователя при помощи произвольных выражений WHERE и т.д.

Дополнительная информация:

- домашняя страница модуля mod_sql: *http://www.lastditcheffort.org/~aah/proftpd/mod_sql/*;
- домашняя страница proftpd: *http://www.proftpd.org/*.

СОВЕТ №86 Оптимизация glibc, linuxthreads и ядра системы для сервера MySQL

(Убедитесь, что операционная система, на которой установлена ваша база данных, благодаря перечисленным ниже методам работает с максимальной эффективностью)

При чрезвычайно загруженном сервере MySQL, обслуживающем, например, более 800 запросов в секунду и несколько сотен клиентов одновременно, можно столкнуться с ограничениями самой операционной системы, мешающими максимально эффективной работе MySQL. В крайних случаях, на системах с неправильно ведущей себя библиотекой потоков MYSQL может захватывать все доступные процессорные ресурсы, искусственно приводя к 100% загрузке сервера.

Создав выделенный сервер MYSQL и изменив некоторые значения по умолчанию в glibc, linuxthreads и ядре Linux, можно сделать возможным обслуживание одним компьютером тысяч одновременных запросов. Разумеется, необходимо соответствующее аппаратное обеспечение для поддержки такой нагрузки, но благодаря внесенным изменениям операционная система и дистрибутив MySQL перестанут являться сдерживающим фактором для быстродействия.

Внимание! Приводимые далее шаги включают внесение изменений в критичные параметры сервера. Не начинайте модифицировать исходные коды libc и ядра системы, не обладая соответствующими навыками и не имея полной работоспособной резервной копии всей системы, желательно хранимой вдали от основного офиса. Эту процедуру следует выполнять только на выделенном сервере MySQL и только в

том случае, если вы уверены, что все остальное в порядке (особенно это касается переменных в файле */etc/my.cnf*).

Действие 1: Скомпилируйте glib. Загрузите исходные коды утилиты glibc по адресу *http://www.gnu.org/software/libc/libc.html*. На момент написания книги самой свежей версией glibc являлась версия 2.2.5. Кроме того, загрузите копию модуля linuxthreads, подходящую для работы с загруженной версией glibc.

Распакуйте архив glibc и установите модуль linuxthreads, как описано в инструкции по установке дистрибутива glibc. Далее следуйте методикам, описанным на сайте *http://www.mysql.com/doc/en/Linux.html*, включая такие советы:

- измените в файле *sysdeps/unix/sisv/linux/bits/local_lim.h* значение параметра PTHREAD_THREADS_MAX на 4096;
- измените в файле *linuxthreads/internals.h* значение параметра STACK_SIZE на 256 Кб.

Запустите утилиту configure и задайте следующий переключатель: `--prefix=/usr/local/glibc-2.2.5`. С его помощью компиляция утилиты glibc будет произведена в отдельную папку, вместо замены существующей системной glibc. Это важно для запуска команды make install, поскольку попытка перезаписи запущенной системной службы glibc обычно приводит к полной неразберихе в библиотеках! А сломанную libc сложно восстановить без статически скомпилированных системных утилит и загрузочного диска, поэтому не рекомендуем это делать, если только нет в запасе лишнего времени и полной резервной копии только что сломанной системы. Компиляция glibc в альтернативную директорию, например */usr/local/glibc-2.2.5*, позволяет аккуратно обойти проблему обновления системных libc.

Скомпилируйте и установите glibc. Если никаких затруднений не возникнет, в вашем распоряжении окажется самая новая версия glibc, установленная в папку */usr/local/glibc-2.2.5/*.

Действие 2: Ядро. Внесите в дерево исходных кодов ядра следующие изменения:

- в файле *include/linux/limits.h* значение параметра NR_OPEN замените на 4096;
- в файле *include/linux/fs.h* значение параметра INR_OPEN замените на 4096.

Перекомпилируйте и переустановите ядро системы и все модули, после чего перезагрузите систему.

Действие 3: Перекомпилируйте сервер MySQL. Загрузите и распакуйте исходные коды для сервера MySQL. Запустите команду configure

с переключателем --use-other-libc=/usr/local/glibc-2.2.5. Сервер MySQL будет связан с новой glibc, вместо системной. Теперь перекомпилируйте и установите MySQL как обычно.

Действие 4: Увеличьте максимальное количество файловых дескрипторов во время загрузки. Добавьте следующую строку в файл */etc/rc.d/rc.local* (либо в другой файл, включенный в автозагрузку) до запуска safe_mysqld:

```
echo 65536 > /proc/sys/fs/file-max
```

Выполните перезагрузку или запустите вышеупомянутую команду вручную, а затем запустите safe_mysqld. Как показали проведенные нами тесты, максимальное количество допустимых подключений увеличилось с нескольких сотен до 4090 без видимого повышения загрузки процессора. Запуск с аналогичными параметрами сервера, выделенного для работы с базой данных, не вызвал никаких проблем с производительностью либо стабильностью работы.

Если же объемы обмена данными с базой достаточно велики, следует подумать о распределении нагрузки между несколькими серверами, как описано в разделе *«Настройка репликации в MySQL»* **[Совет № 82]**.

СОВЕТ №87

Панель инструментов Apache

(Используйте сценарий инсталляции для автоматической загрузки, настройки, компиляции и установки сервера Apache и его компонентов)

Панель инструментов Apache, автором которой является Брайан Эндрю, предоставляет настраиваемый, управляемый меню интерфейс для загрузки и компиляции сервера Apache, mod_perl, MySQL, PHP и т.д. Она обладает встроенной поддержкой для:

Apache

веб-сервера;

SSL

протокола безопасных сокетов для защищенного обмена данными с веб-сервером;

PHP, mod_perl mod_fastcgi

быстродействующей поддержки языка сценариев;

MySQL

быстродействующего сервера баз данных;

OpenLDAP, mod_auth_ldap, mod_auth_radius, mod_auth_pop, mod_auth_sys и mod_accessref

различных средств аутентификации и авторизации;

WebDAV и mod_layout

простых и в то же время достаточно мощных средств веб-дизайна;

mod_dynvhost, mod_throttle, mod_gzip и mod_bandwith

эффективного хостинга и управления сервером.

Панель инстументов Apache является настраиваемой и обеспечив поддержку любых модулей, которые вы захотите подключить к ней дополнительно. Ведь это, в конце концов, всего лишь сценарий оболочки командной строки.

Панель инструментов выпускается в двух видах: простой сценарий, позволяющий загружать при необходимости исходные коды различных компонентов, и полный пакет, включающий в себя сценарий и все необходимые ресурсы. Панель даже позволяет перехватывать конфликты RPM (менеджера пакетов – Redhat Package Manager) в случае их возникновения.

Панель представляет собой сценарий оболочки, вызываемый из командной строки. Его следует запускать с привилегиями пользователя root для установки различных модулей в нужные места. Первым шагом в управляемом меню графическом интерфейсе является выбор различных пакетов, которые необходимо установить:

```
————————————————————————————————————————————————
Apache Toolbox 1.5.59
Support: http://www.apachetoolbox.com
————————————————————————————————————————————————
[+] apache) Apache submenu...
[-] php) PHP submenu (v4.2.2)...
[-] rpm) Build an RPM with our choices?
[-] page2) Apache Modules PAGE 2 ...
————————————————————————————————————————————————
[-] 1) GD 2.0.1 [-] -SQL DB Menus-
[-] 4) Mod Python 2.7.8 [-] 5) Mod_SSL+OpenSSL
[-] 6) -Mod Throttle 312 [-] 7) -WebDAV 1.0.3-1.3.6
[-] 8) -Mod FastCGI [-] 9) -Mod AuthNDS 0.5
[-] 10) -FrontPage 2002 [-] 11) -Mod GZip 1.3.19.1a
[-] 12) -Mod DynaVHost [-] 13) -Mod Roaming
[-] 14) -Mod AccessRef 1.0.1 [-] 15) -Mod AuthSYS
[-] 16) -Mod Bandwidth [-] 17) -Mod Perl 1.27
[-] 18) -Mod Auth LDAP [-] 19) Apache Jakarta
[-] 20) -Mod Auth Radius [-] 21) -Mod Auth POP3
[-] 22) -Mod Layout 3.2 [-] 23 -Mod DTCL
q) Quit 99) Descriptions
go) Compile selections...
————————————————————————————————————————————————
```

Не уверены, что все эти блага присутствуют на вашей системе? Наберите в командной строке 99<enter&gt для получения подробного описания. Создав список необходимых функциональных возможностей, наберите в командной строке go — инструментальная панель начнет свою работу. Первая остановка: проверка конфликтов менеджера пакетов RPM:

```
————————————————————————————————
————————————-Scanning for RPM's ————————————————
————————————————————————————————
Testing for PHP RPM... not found.
Testing for PHP IMAP RPM... not found.
Testing for GD RPM... not found.
Testing for GD Devel RPM... not found.
Testing for Apache RPM... not found.
...
Testing for OpenLDAP RPM... not found.
Testing for OpenLDAP Devel RPM... not found.
[+] Wget found!
————————————————————————————————
```

Панель инструментов Apache попросит подтвердить путь установки (*/usr/local/apache*), позволяя менять его в соответствии с вашими предпочтениями, после чего работа продолжится.

Если исходные коды (архив *tar.gz*) не будут найдены, сценарий запросит разрешения на их загрузку:

```
[+] Setting up Apache source...
[-] apache_1.3.26.tar.gz detection failed
Do you wish to download it now [y/n] y

--21:35:40--
--ftp://mirrors.partnersforever.net:21/pub/apache/dist/
apache_1.3.26.tar.gz => 'apache_1.3.26.tar.gz'
Connecting to mirrors.partnersforever.net:21... connected!
Logging in as anonymous ... Logged in!
==> TYPE I ... done. ==> CWD pub/apache.dist ... done.
==> PORT ... done. ==> RETR apache_1.3.26.tar.gz ... done.

0K -> ..... ..... ..... ..... [2%]
50K -> ..... ..... ..... ..... [5%]
...
```

Затем панель инструментов выполнит в фоновом режиме настройку, интеграцию, предварительную подготовку и т.д.:

```
[+] Uncompressed Apache source...
[+] Getting apache pre-configured
[+] Apache pre-configured
```

```
[+] Apache http.cond-dist updated for SSI support
[+] Getting GD lib's with PNG and zlib support ready...
...
```

После небольшой паузы будет предоставлена возможность отредактировать сценарий конфигурации Apache, если вы настолько уверены в своих знаниях. Если все будет в порядке, то в конце концов получите знакомый сайт, конечно, при условии, что вы занимались настройкой сервера Apache раньше:

```
Configuring for Apache, Version 1.3.26
+ using installation path layout: Apache (config.layout)
/* используется путь установки: Apache (config.layout)
+ activated php4 module (modules/php4/libphp4.a)
/* активируем модуль php4 (modules/php4/libphp4.a)
+ Warning: You have enabled the suEXEC feature. Be aware
/* Предупреждение: у вас включена функция suEXEC. Учтите,
+ that you need root privileges to complete the final
/* что вам необходимы полномочия пользователя root
+ installation step.
/* для завершающего этапа установки.
Creating Makefile
Creating Configuration.apaci in src
Creating Makefile in src
+ configured for Linux platform
/* настройка для Linux завершена
...
Creating Makefile in src/modules/standard
Creating Makefile in src/modules/php4
[+] Done Configuring Apache source
/* Конфигурирование исходных кодов Apache завершено

-------------------------------------------------------------
При условии отсутствия ошибок, запустите «cd apache_1.3.26;make» прямо сейчас. Запустите команду «make install» в папке с исходными кодами apache для установки apache версии 1.3.26
-------------------------------------------------------------
```

Так что перейдите к директории Apache и выполните команды make, make text и make install!

Дополнительная информация:

- оригинальная статья по данной теме, размещенная на сайте *http://www.onlamp.com/pub/a/apache/2000/11/17/wrangler.html*;
- Панель инструментов Apache;
- «Руководство Linux Apache MySQL PHP (LAMP) (Linux Help.net);
- столбцы HTTP Wrangler (O'Reilly Network).

СОВЕТ №88

Отображение полного имени файла в индексах

(Избавьтесь от усечения имен файлов в автоматически индексируемых директориях)

Вы когда-нибудь обращали внимание на то, что сервер Apache урезает имена файлов в списке директорий? Весьма неприятно зайти на какую-нибудь веб-страницу, полную интересных архивов, и увидеть в результате нечто типа:

```
Index of /~rob/stuff/kernel

Name Last modified Size Description

Parent Directory 03-Sep-2002 00:33 -
patched-linux-2.14.12..> 11-Oct-2001 00:59 22.0M
patched-linux-2.14.12..> 11-Oct-2001 00:59 1K
patched-linux-2.14.12..> 11-Oct-2001 00:59 27.1M
patched-linux-2.14.12..> 11-Oct-2001 00:59 1K
patched-linux-2.14.12..> 23-Oct-2001 22:28 22.0M
patched-linux-2.14.12..> 23-Oct-2001 22:28 1K
patched-linux-2.14.12..> 23-Oct-2001 22:28 27.2M
patched-linux-2.14.12..> 23-Oct-2001 22:28 1K
patched-linux-2.14.12..> 05-Nov-2001 15:30 22.1M
patched-linux-2.14.12..> 05-Nov-2001 15:30 1K
patched-linux-2.14.12..> 05-Nov-2001 15:30 27.4M
patched-linux-2.14.12..> 05-Nov-2001 15:30 1K
patched-linux-2.14.12..> 22-Nov-2001 22:18 22.6M
patched-linux-2.14.12..> 22-Nov-2001 22:18 1K
patched-linux-2.14.12..> 22-Nov-2001 22:18 28.0M
patched-linux-2.14.12..> 22-Nov-2001 22:18 1K
```

Невозможно сказать, какой файл каким является, не наводя указатель мыши на каждую ссылку. Изменить поведение по умолчанию можно при помощи строки IndexOptions в файле *http.conf*, включающей опцию NameWidth:

```
IndexOptions FancyIndexing NameWidth=*
```

По умолчанию, дистрибутив Apache всегда поставляется с FancyIndexing. Теперь выполните команду apachectl restart и перезагрузите страницу, чтобы увидеть полные имена:

```
Index of /~rob/stuff/kernel

Parent Directory 03-Sep-2002 00:33 – Description
patched-linux-2.14.12.tar.bz2 11-Oct-2001 00:59 22.0M
patched-linux-2.14.12.tar.bz2.sign 11-Oct-2001 00:59 1K
patched-linux-2.14.12.tar.gz 11-Oct-2001 00:59 27.1M
patched-linux-2.14.12.tar.gz.sign 11-Oct-2001 00:59 1K
patched-linux-2.14.12.tar.bz2 23-Oct-2001 22:28 22.0M
patched-linux-2.14.12.tar.bz2.sign 23-Oct-2001 22:28 1K
```

```
patched-linux-2.14.12.tar.gz 23-Oct-2001 22:28 27.2M
patched-linux-2.14.12.tar.gz.sign 23-Oct-2001 22:28 1K
patched-linux-2.14.12.tar.bz2 05-Nov-2001 15:30 22.1M
patched-linux-2.14.12.tar.bz2.sign 05-Nov-2001 15:30 1K
patched-linux-2.14.12.tar.gz 05-Nov-2001 15:30 27.4M
patched-linux-2.14.12.tar.gz.sign 05-Nov-2001 15:30 1K
patched-linux-2.14.12.tar.bz2 22-Nov-2001 22:18 22.6M
patched-linux-2.14.12.tar.bz2.sign 22-Nov-2001 22:18 1K
patched-linux-2.14.12.tar.gz 22-Nov-2001 22:18 28.0M
patched-linux-2.14.12.tar.gz.sign 22-Nov-2001 22:18 1K
```

Намного лучше. Установите это значение по умолчанию для своего сервера Apache – и пользователи еще скажут вам «спасибо».

СОВЕТ №89

Быстрое изменение конфигурации при помощи IfDefine

(Вносите изменения в конфигурацию сервера Apache без редактирования файла httpd.conf)

Сервер Apache позволяет быстро и легко конфигурировать себя при помощи директив IfDefine и IfModule. С помощью IfDefine активизируются определенные группы настроек путем задания специальных флажков в командной строке:

```
# /usr/local/apache/bin/httpd -DSSL
```

Это распространенный пример определения (-D) нового флажка SSL. Его дальнейшее включение активизирует следующие группы настроек:

```
<IfDefine SSL>
# любые заданные здесь настройки будут активизированы в случае
# задания в командной строке флажка -DSSL
</IfDefine>
```

Аналогичным образом работает директива IfModule, проверяющая, загружен ли тот или иной модуль в файле настроек Apache, т.е. не закомментирован ли он. Например, если модуль mod_userdir загружен, будет активирована следующая конфигурация; в противном случае эти настройки будут просто проигнорированы:

```
<IfModule mod_userdir.c>
UserDir public.html
</IfModule>
```

С расширением сервера Apache, файл *httpd.conf* начинает приобретать более ярко выраженную модульную структуру, позволяя выполнять тонкую настройку различных модулей, не беспокоясь о порче дистрибутива из-за нарушения зависимостей.

Пользуясь рассмотренными выше примерами, можно легко менять настройки сервера Apache из командной строки. Предположим, что каждую пятницу необходимо отправлять бюллетень десяти тысячам человек. Кроме того, по опыту известно, что наибольшую посещаемость ваш сайт имеет в выходные дни. Воспользуйтесь для настройки сервера Apache следующим фрагментом кода, в котором применяется директива IfDefine:

```
<IfDefine !WEEKEND>
MinSpareServers 1
MaxSpareServers 5
StartServers 1
MaxClients 150
</IfDefine>

<IfDefine WEEKEND>
MinSpareServers 5
MaxSpareServers 15
StartServers 5
MaxClients 300
</IfDefine>
```

При такой конфигурации можете запустить свой сервер в нормальном режиме, воспользовавшись настройками по умолчанию для задания начального количества запущенных серверов и обслуживания:

```
# /usr/local/apache/bin/httpd
```

А перед уходом с работы в пятницу можете просто остановить работу демона и перезапустить его с таким параметром:

```
# /usr/sbin/httpd -DWEEKEND
```

После перезапуска сервер начнет работать с настройками, оптимизированными для трафика выходного дня.

Приведем еще один пример использования директивы IfModule:

```
<IfModule libperl.so>
PerlModule Apache::Registry
<Location /usr/local/httpd/cgi-bin>
SetHandler perl-script
PerlHandler Apache::Registry
Options +ExecCGI
</Location>
</IfModule>
```

В этом случае модуль mod_perl будет автоматически активирован при запуске любого из наших сценариев CGI, в какой бы момент времени ни был загружен данный модуль. Использование этой директивы имеет смысл и для предыдущего сценария — благодаря включению мо-

дуля mod_perl они станут оперативнее реагировать на возросший объем трафика. Если же потребуется избавиться на выходные дни от расхода ресурсов, связанных с работой модуля mod_perl или для упрощения отладки ошибок CGI, достаточно будет закомментировать строки LoadModule и AddModule в mod_perl, и все вернется к параметрам по умолчанию.

Можете применить эту идею для создания специфической конфигурации разработчика, вызываемой при помощи флажка –DEVELOPER, или даже конфигурации -VIRTUAL_HOSTS, позволяющей отключать клиентов, не заплативших вовремя за веб-хостинг:

```
<IfDefine VIRTUAL_HOSTS
# Все <VirtualHosts>, которые вы хотите заблокировать, должны быть
# перечислены здесь. Без задания в командной строке флажка -DVIRTUAL_HOSTS
# (означающего, к примеру, что клиенты оплатили положенную сумму)
# все принадлежащие им узлы сети будут отключены
<IfDefine>

<IfDefine EVELOPER>
# Обратите внимание на нашу уловку со словом EVELOPER,
# призванную сделать командную строку более читаемой;
# при задании флажка -DEVELOPER загружаются различные дополнительные
# модули, не используемые в рабочем окружении
LoadModule proxy_module libexec/httpd/libproxy.so
LoadModule expires_module libexec/httpd/mod_expires.so
LoadModule usertrack_module libexec/httpd/mod_usertrack.so
AddModule mod_proxy.so
AddModule mod_expires.so
AddModule mod_usertrack.so
<IfDefine>
```

Если для запуска или отключения сервера Apache на вашей системе используется apachectl, можете отредактировать его в соответствии со своими предпочтениями, т.к. это всего лишь сценарий командной строки. Найдите в файле apachectl следующую строку:

```
HTTPD=/usr/local/apache/bin/httpd
```

и измените ее на нечто типа:

```
HTTPD=«/usr/local/apache/bin/httpd -DSSL -DEVELOPER _DVIRTUAL_HOSTS»
```

Теперь при запуске команды apachectl start задаваемые флаги -D будут корректно передаваться в httpd.

СОВЕТ №90

Упрощенное отслеживание посещаемости по рекламным ссылкам

(Создание статистики посещаемости веб-узла по различным видам рекламы с помощью простой строки QUERY_STRING)

Если вы рассчитываете на рекламу, размещенную в журналах, на веб-сайтах и в газетных изданиях, вам наверняка понадобится информация о том, насколько возросла посещаемость благодаря каждому виду рекламы. Это поможет в дальнейшем более продуманно тратить свои деньги. К сожалению, как всегда, не хватает времени на реализацию более сложных решений, поэтому требуется простое решение, позволяющее легко анализировать полученные результаты.

Сервер Apache с готовностью заносит в журналы значения всех переменных среды, передаваемых на обслуживаемые им веб-страницы. Таким образом, реализовать этот способ предельно просто: нужно для каждого вида рекламы указывать свою разновидность ссылки — добавлять QUERY_STRING. Строка запроса будет просто проигнорирована, если не задано никаких особых действий для ее обработки, например при помощи SSI, PHP, CGI и т.д., но при этом сервер Apache в любом случае сохранит в журнале access_log информацию о строке запроса.

Например, вы поместили объявление в газете *New York Times*. Вместо того чтобы писать в рекламе: «Посетите наш сайт по адресу *http://www.GatesMcFaddenCo.com*», укажите в рекламе строку запроса, которая помогла бы определить, откуда посетитель узнал об этом сайте:

```
http://www.GatesMcFaddenCo.com/?nyt
```

В приведенном примере любой человек, набравший указанный адрес, попадет на главную страницу, но при этом в журнале сервера Apache появится запись о том, что в качестве строки запроса была передана строка nyt. Затем при помощи программы анализа пакетов, например analog либо одной из простых программ, перечисленных ниже, можно подсчитать, сколько людей посетили ваш сайт, прочитав рекламу в *New York Times*.

Применяя QUERY_STRING, можно менять поведение веб-узла по отношению к клиенту в зависимости от строки запроса при помощи любых серверных языков, например, SSI, PHP и т.д. Для примера приведем веб-страницу, выводящую разные приветствия в зависимости от того, какой адрес был набран пользователем.

```
<html>
<head>
<title>Apache Hack #12396 – Simplistic Ad Referrel Tracker</title>
</head>
<body>
```

```
<!—#if expr= «\«$QUERY_STRING\» = \«nyt\»» —>
<h1>Welcome New York Times Reader!</h1>
<!—#if expr=«\«$QUERY_STRING\» = \«xml\»» —>
<h1>Welcome XML.com Reader!</h1>
<!—#else —>
<h1>Welcome To Our Site!</h1>
<!—#endif —>
</body>
</html>
```

Данная страница будет выводить различные строки приветствия в зависимости от того, какую строку запроса QUERY_STRING наберет пользователь: ту, что соответствует рекламе на XML.com, либо ту, что соответствует рекламе в *New York Times*. Если строка запроса не определена, клиент увидит общее приветствие. На некоторых сайтах в зависимости от заданного запроса меняется не только строка приветствия, но и цвета, логотип и даже содержимое рекламных баннеров.

Будьте осторожны, выбирая QUERY_STRING. Если строка запроса *слишком длинна, трудна для запоминания либо выглядит сомнительно*, пользователь может ошибиться в наборе адреса, вызвав неверную статистику, либо вообще не станет набирать QUERY_STRING. В худшем случае потенциальный клиент может вообще отказаться от посещения вашего сайта. Ниже приводятся примеры того, как НЕ надо делать:

```
# эта строка запроса слишком длинная.
http://www.gamegrene.com/?newyorktimes-04-21

# эту строку трудно запомнить и набрать
http://www.gamegrene.com/?04xlmfo3d

# а эта может показаться людям подозрительной
http://www.gamegrene.com/?track=nyt
```

При размещении рекламы в сети строка запроса, закодированная в якорных тегах, может иметь произвольный вид и длину, т.к. пользователю не понадобится ее набирать, достаточно будет щелкнуть по ней мышью. Для анализа полученной статистики воспользуйтесь следующим сценарием на языке Perl:

Листинг: referral-report.pl

```
#!/usr/bin/perl -w
use strict;

my %ads;

# Определим каждую строку QUERY_STRINGS
# и соответствующее ей «реальное имя».
```

```
# Этот сценарий позволяет выполнять поиск только тех
# QUERY_STRINGS, которые состоят из букв и цифр
# (никаких дефисов и символов пунктуации)
$ads{«xml»} = «XML.com»;
$ads{«nyt»} = «New York Times»;
$ads{«ora»} = «»;

# Место размещения журнала доступа сервера Apache.
my $logfile = «/var/log/httpd/access_log»;

# Определим ряд счетчиков.
my %ads_count; my $total;

# Откроем файл журнала
open(LOG, «<$logfile») or die $!;

# и начнем его построчный анализ.
while (<LOG>) {
 # Пропускаем строки, которые нас не интересуют.
 next unless /GET/; next unless /\?(\w+)\s/;
 # Сохраняем строку запроса query_string.
 my $query_string = $1;
 # Продолжаем анализ, если соответствие не найдено,
 # и увеличиваем значение счетчика в противном случае.
 next unless exists $ads{$query_string};
 $total++; $ads_count{$query_string}++;
}

# Выводим информацию на экран.
print «There were a total of $total ad referrals.\n»;
foreach ( sort keys %ads_count ) {
 print «$ads{$_} had $ads_count{$_} ad referrals.\n»;
}

# Закрываем файл журнала и выходим из программы.
close(LOG); exit;
```

Результаты выполнения данного сценария будут выглядеть приблизительно так:

```
There were a total of 17 ad referrals.
/* По рекламным ссылкам сайт посетили 17 человек
XML.com had 6 ad referrals.
/* Из них 6 человек узнали о рекламе с сайта XML.com и т.д.
New York Times had 7 ad referrals.
O'Reilly and Associates had 4 ad referrals.
```

Так как этот код чрезвычайно упрощен, он не позволяет отслеживать многократные обращения с одного и того же IP-адреса либо вычислять процентное соотношение количества обращений, вызванных определенным видом рекламы к общему количеству обращений. Но обе эти функции легко реализуемы человеком хорошо знакомым с языком Perl.

СОВЕТ №91

Имитация FTP-серверов при помощи сервера Apache

(Установка нескольких уровней анонимного доступа на сервере Apache)

Предположим, требуется, чтобы часть веб-сайта работала в качестве FTP-сервера, то есть, например, некоторые пользователи имели учетные записи для загрузки информации, а другие могли загружать ее анонимно. В то же время последние должны обладать правами на загрузку только одного определенного типа файлов. Вы не доверяете своей клиентуре в плане установки и использования программ FTP, поэтому хотите организовать работу через обычный браузер.

Воспользовавшись включенными по умолчанию, но редко применяемыми модулями Apache, можно организовать аутентифицированный анонимный доступ к определенным частям веб-сайта либо вернуться к использованию стандартных методов аутентификации при помощи модуля mod_auth. Это может быть очень удобно, если на сайте хранится большое количество документов, которые не должны индексироваться поисковыми сайтами, либо необходимо создать эффект нескольких уровней привилегий для пользователя: анонимный, привилегированный и доступ для «пиявок» (так называют людей, которые являются исключительно потребителями информационных ресурсов, программного обеспечения и т.д., не делающими никаких вкладов со своей стороны – *прим. перев.*). Причем все это должно быть организовано через обычный браузер без написания каких-либо специальных программ.

Первым делом необходимо включить модуль anon_auth_module, обычно являющийся закомментированным в конфигурационном файле сервера Apache. Для этого нужно найти в этом файле следующие строки (в различных дистрибутивах они могут немного отличаться) и удалить в начале их символ #:

```
#LoadModule anon_auth_module libexec/httpd/mod_auth_anon.so
#AddModule mod_auth_anon.c
```

Убрав символы комментариев из начал строк и перезапустив сервер Apache, можно приступать к добавлению соответствующих директив в файл *httpd.conf* либо в файл *.htaccess*, размещенный в той папке, которую требуется защитить. Предположим, что будет использоваться файл *.htaccess*. Добавьте в него следующие строки и сохраните в защищаемом каталоге:

```
AuthName «anonymous/your email address»
AuthType Basic
Require valid-user

Anonymous orko bender
Anonymous_Authoritative on
```

Первые три директивы используются и при обычной защите доступа к директориям при помощи модуля mod_auth. Директива Anonymous задает имена пользователей, которые должны считаться анонимными. В данном случае речь идет о пользователях orko и bender, но мы легко можем выбрать и другие имена, например, anonymous, anon или guest.

Элемент управления Anonymous_Authoritative определяет, что делать с именами неавторизованных пользователей и паролями — перенаправлять ли их на обработку в другую схему аутентификации. Если мы зададим значение on, анонимность гарантируется только в том случае, когда пользователь подключится под именем orko или bender: в противном случае доступ ему предоставлен не будет.

С другой стороны, если мы зададим в параметре Anonymous_Authoritative значение off, тогда мы сможем обращаться к функциям модуля mod_auth — аутентификации при помощи паролей, групп и т.п. Взгляните на конфигурацию, приведенную ниже. Если пользователь войдет в систему под именем, отличным от heenie или retrogirl, то будет выполнен поиск соответствующей учетной записи в файле, заданном директивой AuthUserFile. Если таковая не обнаружится, доступ для этого пользователя будет запрещен:

```
AuthName "anonymous/your email address"
AuthUserFile /Library/WebServer/.htpasswd
AuthType Basic
Require valid-user

Anonymous heenie retrogirl
Anonymous_Authoritative on
```

Вы можете упростить либо усложнить настройки в соответствии со своими нуждами.

Следующий вариант конфигурации позволяет видеть содержимое каталога любому пользователю (вернее, ту часть содержимого, которая для него доступна). Любой пользователь, упомянутый в файле AuthUserFile, сможет получить доступ ко всем файлам *.jpg* точно так же, как анонимные пользователи, подключившиеся к системе под именами mrs_deception либо spiderj и задавшие действующий адрес электронной почты, содержащий символы «@» и «.»: сервер Apache не проверяет, существует ли такой узел сети в реальности. И наконец, доступ к mp3-файлам сможет получить только пользователь eustate:

```
AuthType Basic
AuthName "anonymous/your email address"
AuthUserFile /Library/WebServer/.htpasswd
```

```
Anonymous mrs_deception spiderj
Anonymous_Authoritative off
Anonymous_VerifyEmail on

<Files *.jpg>
Require valid-user
</Files>

<Files *.mp3>
Require user eustate
</Files>
```

Можно пойти еще дальше, ограничив возможности доступа в зависимости от IP-адреса, имени узла, переменных среды и т.д. Не забудьте только комментировать все изменения — иначе очень легко можете запутаться в том, какие пользователи к каким ресурсам имеют доступ.

К сожалению, невозможно воссоздать такую функциональную возможность FTP-сервера, как выгрузка файлов. Этого не умеет делать ни один из модулей, поставляемых по умолчанию с сервером Apache. Чтобы реализовать эту функцию, понадобится прибегнуть к помощи сценариев CGI, позволяющих осуществлять выгрузку файлов, либо попытаться написать программный код самостоятельно.

СОВЕТ №92

Циклический сдвиг и сжатие журналов сервера Apache

(Используйте сценарий на языке Perl для автоматического сжатия журналов сервера даже при условии включения дополнительных возможностей своего веб-узла)

В этом параграфе предлагается небольшой удобный сценарий, позволяющий автоматически выполнять циклический сдвиг данных в файле журнала и сжимать его при помощи утилиты gzip. Сценарий считывает файл настроек *httpd.conf*, затем переходит к обработке всех подключенных файлов настроек и делает отметку о каждом обнаруженном файле журнала. Затем он переименовывает эти файлы, включая в название текущую дату, перезапускает сервер Apache и выполняет сжатие файлов журнала, в именах которых фигурируют метки даты/времени. К дополнительным функциям относятся: возможность изменения принадлежности к группе, заданной в параметре $gid (с помощью команды chgrps); установка разрешений для этой группы на чтение или запись. Таким образом, дальнейшая обработка файлов журнала может производиться при помощи обычных процессов (не обязательно с правами пользователя root).

Запускайте этот сценарий в конце рабочего дня (либо раз в неделю в зависимости от объема трафика), задав время выполнения в планировщике Linux с целью создания постоянного архива журна-

лов и оберегая ваши текущие журналы от разрастания до неимоверных размеров.

Листинг: logflume.pl

```
#!/usr/bin/perl -w
#
# Сценарий предназначен для выполнения циклического сдвига данных
# в файлах журналов сервера Apache и их архивации с учетом операторов
# Includes в файле httpd.conf и других файлах конфигурации
#
use strict;
$|++;

my $server_root = "/usr/local/apache";
my $conf = «$server_root/conf/httpd.conf»;
my $gid         = "wwwadmin";

my (%logs, %included, @files, @gzip);

my $date = `date +%Y%m%d`; chomp $date;

push @files, $conf;

for $conf (@files) {
open(CONF, "<$conf") || die "Cannot open config file $conf: $!\n";

while (<CONF>) {
chomp;
next if /^(\s+)?#/;

if (/(Transfer|Custom|Error)Log\s+(\S+)(\s+)/i) {
      $logs{$2}++;

} elsif (/^(ResourceConfig|Include)\s+(\S+)/i) {
if(!$included{$2}) {
push @files, $2;
$included{$2}++;
}
}
}

close CONF;
}

for my $logfile (sort keys %logs) {
$logfile = "$server_root/$logfile" unless ($logfile =~ m|^/|);
rename($logfile, "$logfile.$date");
push(@gzip, "$logfile.$date");
}

system("$server_root/bin/apachectl restart");
```

```
for my $logfile (@gzip) {
system("gzip $logfile");

# Установите надлежащие разрешения, чтобы выбранные вами сотрудники
# смогли распаковать эти файлы.
system("chgrp $gid $logfile.gz");
system("chmod 664 $logfile.gz");
}
```

СОВЕТ №93

Генерирование сертификатов SSL и запросов на их подпись

(Создайте ключи SSL, CSR и сертификаты для использования на сервере Apache)

Для использования модулей mod_ssl и Apache_ssl на сервере Apache необходимо иметь сертификат подлинности, подписанный надежным центром сертификации. В этом примере мы генерируем сертификат для использования на веб-узле *https://propaganda.discordia.eris/*. Чтобы сгенерировать ключ в OpenSSL, необходимо:

```
hagbard@fnord:~/certs$ openssl genrsa 512/1024 \
> propaganda.discordia.eris.key
warning, not much extra random data, consider using the -rand option
/* предупреждение: недостаточно случайные данные,
/* воспользуйтесь параметром -rand
Generating RSA private key, 512 bit long modulus
/* Генерация закрытого ключа RSA, 512 бит абсолютного значения
..++++++++++
...++++++++++++
e is 65537 (0x10001)
```

В результате выполнения этого программного кода будет создан только закрытый ключ, а не сертификат. Если хотите дополнительно защитить этот ключ с помощью пароля, задайте в командной строке опцию -des3:

```
hagbard@fnord:~/certs$ openssl genrsa -des3 512/1024 \
> propaganda.discordia.eris.key
warning, not much extra random data, consider using the -rand option
/* предупреждение: недостаточно случайные данные,
/* воспользуйтесь параметром -rand
Generating RSA private key, 512 bit long modulus
/* Генерация закрытого ключа RSA, 512 бит абсолютного значения
..++++++++++
...++++++++++++
e is 65537 (0x10001)
Enter PEM pass phrase:
Verifying password – Enter PEM pass phrase:
```

Пароль придется вводить каждый раз при перезапуске сервера Apache, что может быть весьма неудобно при выполнении регулярного

обслуживания: например, для выполнения циклического сдвига данных в файлах журналов http. Но это неудобство уравновешивается потенциальным ущербом, который может быть нанесен в случае получения злоумышленником доступа к этому ключу. Кстати, учтите: если забудете парольную фразу, восстановить ее практически невозможно, поэтому храните ее в надежном месте!

Далее необходимо сформировать запрос на подпись сертификата (Certificate Signing Request) и отправить его в заслуживающий доверия центр сертификации, например Thawte/VeriSign. Наберите в командной строке все, что выделено ниже жирным шрифтом, подставляя в нужных местах свои параметры (после символов /* следуют комментарии на русском языке к выводимой на экране информации):

```
hagbard@fnord:~/certs$ openssl req -new -key propaganda.discordia.eris.key \
> propaganda.discordia.eris.csr

Using configuration from /usr/local/ssl/openssl.cnf
/* Используем конфигурацию из файла /usr/local/ssl/openssl.cnf
You are about to be asked to enter information that will be incorporated into
/* Введите информацию, которая будет включена
your certificate request.
/* в ваш запрос на сертификацию.
What you are about to enter is what is called a Distinguished Name or a DN.
/* Введите отличительное имя (DN)
There are quite a few fields but you can leave some blank
/* Здесь немало параметров, но некоторые из них задавать необязательно
For some fields there will be a default value,
/* Для ряда полей существуют значения по умолчанию
If you enter '.', the field will be left blank.
/* Если вы введете символ '.', значение поля будет оставлено пустым

Country Name (2 letter code) [AU]: US
State or Province Name (full name) [Some-State]: Texas
Locality Name (eg, city) []:Mad Dog
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Discordia, Inc.
Organization Unit Name (eg, section) []:Operations
Common Name (eg, YOUR name) []:propaganda.discordia.eris
Email Address []: norton@discordia.eris
Please enter the following 'extra' attributes
/* Пожалуйста, введите дополнительные атрибуты,
to be sent with your certificate request
/* которые будут отправлены вместе с запросом на сертификацию
A challenge password []:
An optional company name []:
```

Теперь можете взяться за создание самого сертификата. Чтобы воспользоваться услугами Центра сертификации, необходимо отправить ему на подпись файл *.csr*. Так как эта процедура требует времени, вы можете самостоятельно подписать свой сертификат и использовать его в таком виде, пока будет обрабатываться запрос на подпись сертификата:

```
hagbard@fnord:~/certs$ openssl req -x509 \
-key propaganda.discordia.eris.key \
-in propaganda.discordia.eris.csr \
> propaganda.discordia.eris.crt

Using configuration from /usr/local/ssl/openssl.cnf
hagbard@fnord:~/certs$
```

Если вы намереваетесь создать свой собственный Центр сертификации, используйте для подписи сертификата собственный ключ.

Дополнительная информация:

- *«Создание собственного Центра сертификации»* **[Совет № 94]**;
- домашняя страница утилиты OpenSSL, размещенная по адресу *http://www.openssl.org/*.

СОВЕТ №94

Создание собственного Центра сертификации

(Создайте собственный Центр сертификации для подписи своих и чужих сертификатов SSL)

Хорошо известные Центры сертификации, такие как Thawte/VeriSign, предназначены для аутентификации авторитетных, заслуживающих доверия сторонних компаний. Эти центры занимаются подписью сертификатов SSL, которые используются веб-узлами, работающими с чувствительной информацией – номерами кредитных карточек, паролями и т.д. Если сертификат SSL данного веб-узла подписан заслуживающим доверия центром, всегда можно проверить и идентифицировать сервер путем передачи мандатов сертификата. Чтобы Центр сертификации подписал ваш сертификат, необходимо однозначно доказать, что вы не только являетесь тем, за кого себя выдаете, но и обладаете правами на использование сертификата указанным способом. К примеру, я могу доказать центру сертификации, что я действительно Роб Фликенгер, но вряд ли он подпишет мне сертификат на имя Microsoft Corporation, поскольку я не имею никаких прав на использование этого имени.

Утилита OpenSSL отлично подходит для генерирования всего, что требуется для организации собственно центра сертификации. Утилита CA.pl позволяет максимально упростить данный процесс.

В приводимых ниже примерах придется вручную набирать все, что выделено жирным шрифтом, и в нужных местах вводить пароли, которые не отображаются на экране. Итак, чтобы основать собственный Центр сертификации, вам нужно:

```
hagbard@fnord:~/certs$ /usr/local/ssl/misc/CA.pl -newca
CA certificate filename (or enter to create)
```

```
/* Введите имя сертификата Центра сертификации (либо нажмите enter для
/* создания нового

Making CA certificate ...
Using configuration from /usr/local/ssl/openssl.cnf
/* Используем конфигурацию из файла /usr/local/ssl/openssl.cnf
Generating a 1024 bit RSA private key
/* Генерируем закрытый ключ RSA длиной 1024 бита
.....++++++
.....++++++
writing new private key to './demoCA/private/cakey.pem'
/* идет запись нового закрытого ключа в './demoCA/private/cakey.pem'
Enter PEM pass phrase:
/* Введите пароль PEM:
Verifying password - Enter PEM pass phrase:
/* Подтверждение пароля - введите пароль PEM еще раз:
---
You are about to be asked to enter information that will be incorporated
/*Введите информацию, которая будет включена
into your certificate request.
/* в ваш запрос на сертификацию.
What you are about to enter is what is called a Distinguished Name or a DN.
/* Введите отличительное имя (DN)
There are quite a few fields but you can leave some blank
/* Здесь немало параметров, но некоторые из них задавать необязательно
For some fields there will be a default value,
/* Для ряда полей существуют значения по умолчанию
If you enter '.', the field will be left blank.
/* Если вы введете символ '.', значение поля будет оставлено пустым
---
Country Name (2 letter code) [AU]: US
State or Province Name (full name) [Some-State]: California
Locality Name (eg, city) []:Sebastopol
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Illuminatus Enterpises, Ltd
Organization Unit Name (eg, section) []:Administration
Common Name (eg, YOUR name) []:Hagbard Celine
Email Address []:hagbardceline1723@yahoo.com
```

Вы можете гордиться, так как теперь владеете своим собственным Центром сертификации. Оценим ситуацию:

```
hagbard@fnord:~/certs$ ls
demoCA/
hagbard@fnord:~/certs$ cd demoCA/
hagbard@fnord:~/certs/demoCA$ ls -l
total 24
-rw-r--r-- 1 rob users 1407 Sep 8 14:12 cacert.pem
drwxr-xr-x 2 rob users 4096 Sep 8 14:12 certs/
drwxr-xr-x 2 rob users 4096 Sep 8 14:12 crl/
-rw-r--r-- 0 rob users 0 Sep 8 14:12 index.txt
drwxr-xr-x 2 rob users 4096 Sep 8 14:12 newcerts/
drwxr-xr-x 2 rob users 4096 Sep 8 14:12 private/
-rw-r--r-- 1 rob users 3 Sep 8 14:12 serial
```

Открытый ключ для вашего Центра сертификации хранится в файле cacert.pem, а закрытый – в файле *private/cakey.pem*. И теперь вы сможете использовать этот закрытый ключ для подписи других сертификатов SSL.

Чтобы ваш Центр сертификации смог подписывать сертификаты SSL, потребуется создать новый сертификат, который сможет использовать новый веб-сервер (типа Apache). Во-первых, необходимо сгенерировать закрытый ключ и запрос на сертификацию, как было рассказано в разделе *«Генерирование сертификата SSL и запросов на их подпись»* **[Совет № 93]**. Теперь можете удовлетворять подобные запросы при помощи ключа от своего собственного Центра сертификации:

```
hagbard@fnord:~/certs$ openssl ca -policy policy_anything \
-out propaganda.discordia.eris.crt \
-infiles propaganda.discordia.eris.csr
Using configuration from /usr/local/ssl/openssl.cnf
Enter PEM pass phrase:
Check that the request matches the signature
/* Проверяем соответствие запроса подписи
Signature ok
The Subjects Distinguished Name is as follows
countryName :PRINTABLE:'US'
stateorProvinceName :PRINTABLE:'Texas'
localityName :PRINTABLE:'Mad Dog'
organizationName :PRINTABLE:'Discordia, Inc.'
organizationUnitName :PRINTABLE:'Operations'
commonName :PRINTABLE:'propaganda.discordia.eris'
emailAddress :PRINTABLE:'hail@discordia.eris'
Certificate is to be certified until Sep 8 22:49:26 2003 GMT (365 days)
/* Сертификат будет являться действительным до: Sep 8 22:49:26 2003 GMT (365
/* дней). Подписать?
Sign the cerificate? [y/n]y

1 out of 1 certificate request certified, commit? [y/n] y
/* 1 из 1 запросов на подпись сертификата удовлетворен, сохранить изменения?
Write out database with 1 new entries
/* В базу данных добавлена одна новая запись
Data Base updated
/* База данных обновлена
```

Чтобы воспользоваться сертификатом и ключом на сервере Apache при условии использования модуля mod_ssl (либо Apache-ssl), установите эти файлы как обычно, путем задания следующих строк:

```
SSLCertificateFile /usr/local/apache/conf/ssl.crt/propaganda.discordia.eris.crt
SSLCertificateKeyFile /usr/local/apache/conf/ssl.key/propaganda.discordia.eris.key
```

Все это, конечно, интересно, но что произойдет на самом деле, когда клиент попытается подключиться к сайту *https://propaganda.discordia.eris/?* Не сообщит ли браузер клиента об ошибке по причине того, что ему не известен центр сертификации, которым подписан данный

сертификат SSL? Разумеется, так и произойдет, если только вы заранее не установите на системе клиента открытый ключ вашего Центра сертификации. Как это сделать — описано в следующем параграфе.

Дополнительная информация:

- оперативное руководство (вызываемое командой man) для CA.pl;
- домашняя страница утилиты OpenSSL, размещенная по адресу *http://www.openssl.org/*;
- *http://www.cert.org/advisories/CA-2001-04.html.*

СОВЕТ №95

Распространение сертификатов вашего Центра на клиентских браузерах

(Установите сертификаты своего Центра сертификации на клиентском браузере с помощью щелчка кнопкой мыши)

Существует два формата сертификатов, поддерживаемых современными браузерами: pem и der. В ранних версиях Netscape поддерживался только формат pem, тогда как в последних версиях допустимыми являются оба формата. С Internet Explorer ситуация противоположная: в ранних версиях IE поддерживался только формат der, теперь же поддерживаются оба формата. Остальные браузеры поддерживают, как правило, оба формата. Сертификат формата der можно сгенерировать из имеющегося в вашем распоряжении сертификата формата pem при помощи команды в openssl:

```
hagbard@fnord:~/certs$ openssl x509 -in demoCA/cacert.pem \
-outform DER -out cacert.der
```

Кроме того, нужно будет добавить в файл *conf/mime.types* дистрибутива Apache следующую строку:

```
applicaton/x-x509-ca-cert der pem crt
```

Чтобы изменения вступили в силу, необходимо перезапустить сервер Apache. С этого момента можете помещать файлы *cacert.der* и *demoCA/cacert.pem* в любое место на своем веб-сервере и заставлять клиентов устанавливать новый сертификат по нажатию единственной ссылки.

Во время загрузки сертификата от нового Центра в браузере должно появиться диалоговое окно, запрашивающее разрешение на продолжение операции. Если вы решите доверять Центру сертификации, в дальнейшем все сертификаты SSL, подписанные вашим Центром сертификации, будут приниматься без вывода предупреждений для пользователей, как показано на рисунке 8-1:

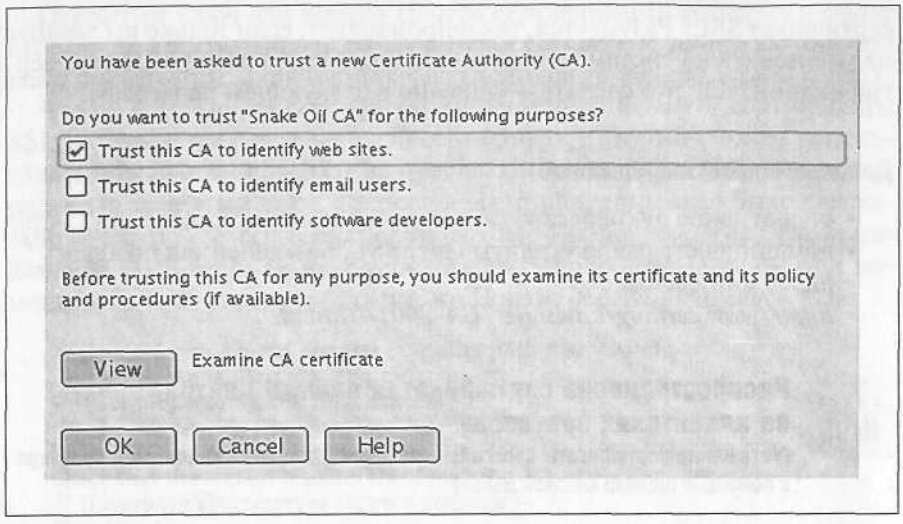

Рис. 8-1. Нажмите OK, чтобы разрешить использование нового Центра сертификации, или View, чтобы прочитать сведения о нем

Помните, что доверять центрам сертификации вслепую нельзя. Разрешайте своему браузеру доверять сертификатам нового центра сертификации, только если вы действительно уверены в том, что он заслуживает доверия: ведь зловредный менеджер данного Центра может подписывать сертификаты любого содержания, которым вы никогда бы не стали доверять, а браузер даже не предупредит вас об этом, т.к. доверяя этому Центру сертификации, вы автоматически соглашаетесь доверять всем подписанным сертификатам. Будьте чрезвычайно осторожны в том, кому можно доверять при использовании браузеров с SSL. Советуем взглянуть на список центров сертификации, перечисленных в настройках браузера, чтобы выяснить, каким доверяет ваш браузер по умолчанию.

Например, известно ли вам, что компания AOL/Time Warner имеет свой собственный Центр сертификации? Как насчет GTE? Или VISA? Сертификаты этих центров поставляются вместе с браузером Netscape 7.0 для Linux и используются для веб-сайтов, электронной почты и дополнительных подключаемых модулей к приложению по умолчанию. Помните об этом при посещении сайтов, использующих безопасное содержимое через SSL: если онлайновое содержимое какого-либо веб-узла подписано Центром сертификации, которому ваш веб-браузер доверяет по умолчанию, тогда при посещении подобного сайта от оператора ПК даже не будет затребовано подтверждение (см. рисунок 8-2).

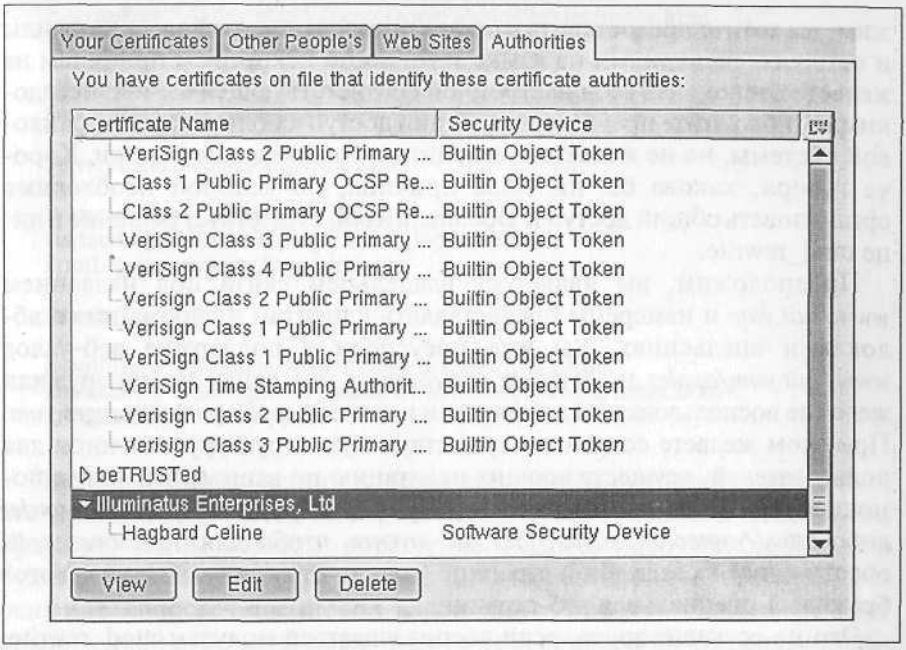

Рис. 8-2. Определение центров сертификации, которые браузер считает заслуживающими доверия

Если вы цените безопасность своего браузера и, соответственно, своей клиентской системы, не забудьте просмотреть список авторитетных центров сертификации.

Дополнительная информация:

- часто задаваемые вопросы по OpenSSL, размещенные по адресу *http://www.openssl.org/support/faq.cgi*;
- *«Генерирование сертификатов SSL и запросов на их подпись»* **[Совет № 93]**;
- *«Создание собственного Центра сертификации»* **[Совет № 94]**.

СОВЕТ №96

Обслуживание нескольких веб-узлов при помощи одного DocumentRoot

(Творческий подход к использованию mod_rewrite позволяет нескольким веб-узлам иметь общий доступ к одному DocumentRoot и при этом выглядеть независимыми)

В некоторых случаях для группы веб-узлов может понадобиться отображение одного и того же корневого каталога документов (DocumentRoot), но при наличии уникальных заглавных страниц. Предполо-

жим, вы хотите предоставить в общее пользование графические файлы и папку со сценариями на языке Java, но по некоторым причинам не желаете беспокоить себя настройкой соответствующей записи псевдонима. Либо хотите предоставить общий доступ к большей части файловой системы, но не желаете применять символические ссылки. Короче говоря, какова бы ни была причина, по которой необходимо организовать общий доступ к DocumentRoot, существует решение в лице mod_rewrite.

Предположим, вы являетесь владельцем сайта под названием *www.fruit.yum* и намерены предоставлять клиентам информацию о яблоках и апельсинах. Вы уже преуспели в поддержке веб-узлов *www.fruit.yum/apples* и *www.fruit.yum/oranges*, но однажды обнаружили желание воспользоваться доменными именами *apples.yum* и *oranges.yum*. При этом желаете сохранить существующую структуру каталогов для пользователей, осуществляющих навигацию по вашему веб-узлу с помощью новых доменных имен (например *http://www.apples.yum/order* либо *http://www.oranges.yum/faq*), но хотите, чтобы для пользователей, обратившихся к заглавной странице (то есть *http://www.apple.yum/*) отображалась специальная веб-страница.

Это не составит труда, если воспользоваться модулем mod_rewrite. Предположим, что уже существует запись VirtualHost под названием *www.apple.yum*:

```
<VirtualHost *>
ServerName www.fruit.yum

ServerAdmin webmaster@fruit.yum

DocumentRoot /home/www/htdocs
CustomLog /home/www/logs/access_log combined
ErrorLog /home/www/logs/error_log
</VirtualHost>
```

Теперь добавьте записи VirtualHost для своих новых доменов, определив в них дополнительные правила с помощью директивы RewriteRule:

```
<VirtualHost *>
ServerName www.apples.yum

ServerAdmin webmaster@fruit.yum

DocumentRoot /home/www/htdocs
CustomLog /home/www/logs/access_log combined
ErrorLog /home/www/logs/error_log

RewriteEngine On
```

```
RewriteRule ^/$ /home/www/htdocs/apples/index.html
RewriteRule ^/index.html$ /home/www/htdocs/apples/index.html
</VirtualHost>

<VirtualHost *>
ServerName www.oranges.yum

ServerAdmin webmaster@fruit.yum

DocumentRoot /home/www/htdocs
CustomLog /home/www/logs/access_log combined
ErrorLog /home/www/logs/error_log

RewriteEngine On

RewriteRule ^/$ /home/www/htdocs/oranges/index.html
RewriteRule ^/index.html$ /home/www/htdocs/oranges/index.html
</VirtualHost>
```

Поскольку все операции перенаправления выполняются локально, это никак не отразится на строке URL, отображаемой в клиентском браузере. Таким образом, обращение к адресу *http://www.oranges.yum* будет выглядеть так, будто клиент зашел на независимый веб-сайт. Обращение непосредственно по адресу *http://www.fruit.yum/oranges* даст аналогичный результат, поскольку во всех ваших html-файлах используются относительные ссылки (либо соответствующие строки псевдонимов). Если хотите реализовать более общее решение, позволяющее работать со столькими адресами www.*.yum, сколько не поленитесь зарегистрировать, можете прибегнуть к следующему примеру:

```
<VirtualHost *>
ServerName www.fruit.yum

ServerAdmin webmaster@fruit.yum

DocumentRoot /home/www/htdocs
CustomLog /home/www/logs/access_log combined
ErrorLog /home/www/logs/error_log

RewriteCond %{HTTP_HOST} www.(.*).yum
RewriteCond %{HTTP_HOST} !www.fruit.yum
RewriteRule (.*) $1 [E=SITE:%1]

RewriteCond %{REQUEST_URI} ^/index.html$
RewriteCond /home/www/htdocs/%{ENV:SITE}/index.html -f
RewriteRule .* /home/www/htdocs/%{ENV:SITE}/index.html

RewriteCond %{REQUEST_URI} ^/$
RewriteCond /home/www/htdocs/%{ENV:SITE}/index.html -f
RewriteRule .* /home/www/htdocs/%{ENV:SITE}/index.html
</VirtualHost>
```

В этом варианте сценария вырезается средняя часть доменного имени (между www. и .yum), после чего проверятся, существует ли файл *index.html* в каталоге с таким именем внутри каталога DocumentRoot данного веб-узла. Если «да», то пользователь перенаправляется к данному файлу.

При работе с mod_rewrite могут возникать проблемы. Оказать помощь в их отладке может директива RewriteLog. Если вы застряли на написании правила RewriteRule, попробуйте добавить следующие строки для виртуального узла сети:

```
RewriteLog /tmp/rewrite.log
RewriteLogLevel 9
```

В результате журнал операций модуля mod_rewrite будет сохраняться в заданном вами файле. Следить за файлом журнала удобно при помощи команды tail -f /tmp/rewrite.log или less -f /tmp/rewrite.log, поскольку в браузере запрашиваются различные URL. Не включайте эту директиву на длительное время на рабочем сервере, поскольку в журнале будут фиксироваться все операции перенаправления, что приведет к замедлению работы сервера Apache.

СОВЕТ №97

Передача содержимого на основе строки запроса при помощи mod_rewrite

(Контроль передачи содержимого на основе строки запросов без помощи сценариев CGI)

Очень удобно использовать строку запросов в строке URL (запросом считается все, что следует после первого символа ?) для выбора содержимого, которое должно быть передано клиенту сервером Apache. Хотя строка запросов чаще всего используется приложениями CGI для считывания и записи программных переменных, она также может использоваться модулем mod_rewrite без привлечения внешних сценариев.

Предположим, у вас имеется система публикаций, которая позволяет делить большие статьи на отдельные страницы. Обычно передача таких страниц организуется с помощью сценария, принимающего в качестве параметра page= номер страницы в строке запросов и отправляющего клиенту страницу с этим номером. Но если вы сохраните локальную копию каждой страницы в файловой системе, для предоставления доступа к кэшированным страницам может использоваться директива RewriteRule, избавляя вас таким образом от накладных расходов, связанных с выполнением сценария при каждом обращении к серверу.

Итак, нам необходимо отправить клиенту второю страницу статьи по адресу *http://mysite.com/news/article.html?page=2*. Задайте следующие правила:

```
RewriteCond %{QUERY_STRING} page=([0123456789]+)
RewriteCond /home/www/htdocs/%{REQUEST_URI}.%1 -f
RewriteRule .* /home/www/htdocs/%{REQUEST_URI}.%1 [L]
```

Если файл *http://mysite.com/news/article2.html* существует, то клиент сможет получить к нему доступ путем внутреннего перенаправления без запуска сценария публикации. Если же файл не существует, тогда обработка запроса будет производиться согласно последнему правилу, путем обычного метода передачи публикаций, скорее всего, с помощью сценария в папке */news*, определенного в параметре ScriptAlias. Этот прием работает для любого количества страниц, укладывающегося в диапазон значений типа integer.

Отсутствие строки запросов может рассматриваться как отдельный случай. Предположим, что сценарий передачи содержимого зависит от целого ряда переменных, но если ни одна из них не задана, тогда клиенту возвращается предварительно определенное содержимое, например, первая страница статьи, имеющая представление по умолчанию. Это еще один пример того, как разумное использование правил RewriteRule позволяет избежать ненужного расхода ресурсов при динамическом генерировании содержимого:

```
RewriteCond %{QUERY_STRING} = ""
RewriteCond /home/www/static/%{REQUEST_URI}.%1 -f
RewriteRule .* /home/www/static/%{REQUEST_URI}.%1 [L]
```

В данном случае мы считаем, если строка запроса существует, то генерация динамического содержимого должна выполняться обязательно. В противном случае вначале выполняется проверка наличия запрошенного содержимого — в папке */home/www/static/* — и, если нужная страница существует, она возвращается клиенту. Если же локальная копия этой страницы отсутствует, обработка запроса осуществляется согласно следующему набору директив (скорее всего, путем выполнения сценария генерации динамического содержимого). Пока устаревание содержимого в кэше регулирует внешний процесс (планировщик либо демон, специально предназначенный для наблюдения за изменениями в содержимом), кэширование, с точки зрения пользователя, будет проходить гладко. Чтобы обновить содержимое, достаточно удалить старые файлы из кэша, и при повторном обращении обновленные страницы будут сгенерированы динамически.

Можно заставить сервер Apache выполнять любые нужные действия на основе строки запроса. Функциональные возможности модуля mod_rewrite ограничены только воображением и сложностью регулярных выражений.

Дополнительная информация:

- *«Распределение нагрузки при помощи директивы RewriteMap сервера Apache»* **[Совет № 99]**;
- *«Обслуживание узлов: массовый веб-хостинг при помощи подстановочных символов, прокси-серверов и перенаправления»* **[Совет № 100]**;
- руководство по модулю mod_rewrite сервера Apache, расположенное по адресу *http://httpd.apache.org/docs/mod/mod_rewrite.html*.

СОВЕТ №98

Применение mod_proxy для ускорения работы сервера Apache

(Перенаправление сложных динамических запросов на другой сервер Apache или на другой компьютер)

Огромные усилия были потрачены на оптимизацию сервера Apache таким образом, чтобы он мог извлекать файлы из файловой системы и передавать их пользователю в ответ на входящие запросы http с максимальной быстротой. Увы, веб-узлы, которые целиком основаны на использовании хранимых файлов, редко пользуются большой популярностью. Высокий спрос на интерактивное содержимое дал толчок рождению многих проектов, специально разработанных для предоставления конечному пользователю динамического содержимого, настраиваемого в широких пределах.

К сожалению, с повышением интерактивности веб-узлов часто страдает производительность. Интерактивное содержимое требует интерпретации языков программирования, а процесс интерпретации произвольного программного кода является чрезвычайно ресурсоемким по сравнению с предоставлением доступа к статическим файлам. Одним из классических примеров «худшего варианта развития событий» считается сценарий Perl CGI, выполняющий запрос к базе данных и отправляющий ответ по электронной почте с помощью внешней программы sendmail. По сравнению с тем временем, которое понадобилось бы для обслуживания обращения к статическому файлу, выполнение одного запроса CGI займет целую вечность, что потребует целой вечности и для обслуживания всех остальных запросов к данному серверу, тормозя работу всего сервера.

Модули Apache, такие как mod_perl и mod_php, при участии вашего программистского ума могут значительно облегчить задачу порождения внешних процессов при каждом обращении. Но даже при помощи интерпретаторов, встроенных в сам сервер Apache, количество динамических запросов, обслуживаемых за единицу времени, редко даже приближается к количеству запросов, выполняемых за то же самое время на статическом сервере. Одна из причин, по которой это происходит, заключается в том, что большинство запросов страдает от потери произво-

дительности, связанной с поддержкой вложенных языков программирования, независимо от того, является запрос динамическим или нет.

Рассмотрим ситуацию с использованием большой инсталляции с модулем mod_php. В результате работы системы в течение всего нескольких минут, один только процесс httpd может поглотить от 40 до 50 мегабайт ОЗУ для кэширования в память модулей языка Perl с целью ускорения выполнения программного кода. Теперь представьте себе поступление запроса к однопиксельной прозрачной картинке в формате gif (такой прием часто используется веб-дизайнерами для создания на веб-странице пустых мест произвольного размера с точностью до пикселя). В самом худшем случае, когда все процессы httpd окажутся заняты, сервер Apache попытается породить еще один процесс. В результате для обслуживания единственного статического запроса к файлу размером в несколько байт сервер будет вынужден вызвать модуль mod_perl и породить для него дочерний процесс, который может использовать до 50 мегабайт ОЗУ и много процессорного времени. Очевидно, что организовав вызов модуля mod_perl только в случае крайней необходимости, можно избавиться от грандиозного перерасхода ресурсов, когда кто-либо обращается к графическим либо другим файлам, не нуждающимся в динамической обработке.

Как вариант, рассмотрите запуск двух серверов Apache: одного работающего со всеми модулями (bell, whistle, vibra-slap), в которых нуждается система управления динамическим содержимым, другого — с максимально урезанным набором функций. Тогда если все запросы будут поступать на облегченную версию сервера Apache, он сможет обслуживать статические запросы обычным способом, а все динамические запросы будут перенаправляться на второй сервер, который и займется их обработкой.

Предположим, у вас запущена облегченная версия сервера Apache, работающая через стандартный порт 80, и сервер приложений через порт 8088. Воспользовавшись приведенным ниже правилом rewrite, можно организовать перенаправление запросов к любому содержимому, за исключением графических файлов и архивов, на сервер приложений. Поместите следующий фрагмент кода в конце записи для VirtualHost своего веб-узла (на сервере, работающем через порт 80):

```
RewriteEngine On

RewriteCond %{REQUEST_URI} !.*\.(jpg|gif|pdf|png|zip|tgz|gz)$
RewriteRule ^/(.*> http://%{HTTP_HOST}:8088/$1 [P]
```

Символ [P] в конце строки правила RewriteRule задает использование прокси. То есть, облегченная версия Apache выполняет подключение к динамическому серверу, работающему с портом 8088, и передает ему по-

ступивший от клиента запрос. После получения ответа облегченный сервер передает ответ клиенту так, как если бы этот запрос был им обслужен самостоятельно. Что касается клиента, то для него ситуация выглядит так, словно он имеет дело с единственным быстродействующим веб-сервером, работающим с портом 80. Этот метод будет прекрасно работать, если на обоих серверах веб-узел указывает на один и тот же DocumentRoot.

Если ваше приложение использует файлы «cookie», понадобится добавить в запись VirtualHost — до правил RewriteRule — еще одну строку с параметром ProxyPassReverse:

```
ProxyPassReverse / http://%{HTTP_HOST}:8088/
```

В результате при использовании прокси сервер Apache будет транслировать заголовки http, чтобы файлы cookie могли передаваться от сервера приложений к клиенту должным образом.

При наличии соответствующего аппаратного обеспечения можно распределить нагрузку между двумя компьютерами, используя один из них в качестве прокси-сервера, а другой — в роли сервера приложений. В таком случае правила должны выглядеть так:

```
ProxyPassReverse / http://your.application.server.here/

RewriteEngine on

RewriteCond %{REQUEST_URI} !.*\.(jpg|gif|pdf|png|zip|tgz|gz)$
RewriteRule ^/(.*> http://your.application.server.here/$1 [P]
```

Разумеется, при использовании нескольких физических серверов придется позаботиться о синхронизации файловой системы (см. раздел *«Синхронизация отдельных частей файловой системы при помощи утилиты rsync»* **[Совет № 41]**), иначе ваши клиенты столкнутся с конфликтующими страницами при каждом обновлении сайта. Если даже два сервера окажутся не в состоянии обслужить все запросы к динамическому содержимому, прочтите раздел *«Распределение нагрузки при помощи директивы RewriteMap сервера Apache»* **[Совет № 99]**, в котором описан один из методов распределения загрузки между неограниченным количеством серверов приложений.

СОВЕТ №99

Распределение нагрузки при помощи директивы RewriteMap сервера Apache

(Распределите нагрузку между произвольным количеством серверов приложений при помощи RewriteMap)

Как было показано в предыдущем разделе, существует довольно прозрачный метод обслуживания содержимого произвольным веб-

сервером с помощью модулей mod_proxy и mod_rewrite. Вместо того чтобы использовать внешнее перенаправление [R], целевой прокси [P] выдает содержимое, возвращенное другим сервером, за данные, поступившие с того сервера, на который клиент отправил оригинальный запрос, причем URL в адресной строке браузера остается без изменений. Одной из областей применения этой технологии является распределение нагрузки по обслуживанию страниц между двумя компьютерами, делая возможным обработку большего количества обращений, чем в случае с единственным компьютером.

Но что если придется обслуживать такие объемы трафика, с которыми не в состоянии справиться единственный сервер приложений? В таком случае понадобится реализовать механизм выбора прокси-серверов из набора доступных так, чтобы иметь возможность перенаправлять большее количество запросов на более мощный сервер. Подобную функциональность предлагает директива RewriteMap.

Синтаксис задания директивы RewriteMap показан ниже:

```
RewriteMap server rnd:/usr/local/apache/con/servers.map
```

Она просто описывает список вызываемых серверов, сохраняемый в файле *servers.map* внутри директории *conf/*. Функция rnd: позволяет делать случайный выбор из перечня.

Формат файла *servers.map* очень прост:

```
web www1|www2|www3|www4|www5
```

Слева задается имя переменной (web), которая может использоваться практически в любых файлах конфигурации, например в правиле RewriteRule:

```
RewriteRule ^/(.*) http://${server:web}.oreillynet.com/$1 [P]
```

Справа задается список имен серверов, возвращаемых в произвольном порядке. В качестве разделителей используются вертикальные линии. В данном примере с правилом RewriteRule для подстановки значения переменной ${server:web} из файла *servers.map* в произвольном порядке будет выбираться одно из значений — www1, www2, www3, www4 или www5. В результате прокси-сервер будет перенаправлять клиента на сервер *http://www4.oreillynet.com* (или же любой другой из доступного списка серверов www*) при задании им обычного URI.

Единственная проблема при использовании полностью случайного списка узлов состоит в том, что все перечисленные серверы будут принимать на себя приблизительно равные доли трафика. Но если сервер www1 представляет собой четырехпроцессорную систему на базе Xeon 2.4ГГц с 8Гб ОЗУ, а www2 — компьютер 486SX/33 с 4Мб ОЗУ, придет-

ся подумать о том, чтобы перенаправлять на www1 больший объем трафика, чем на сервер www2. С этой целью достаточно перечислить более мощные компьютеры в карте большее число раз:

```
web www1|www1|www1|www1|www2|www3|www3|www4|www5|www5
```

В данном примере сервер www1 будет обслуживать 4 из 10 обращений, www2 — 1 из 10 обращений, www3 — 2 из 10, www4 — 1 из 10 и www5 — 2 из 10. Можно выполнить более точную настройку, запустив утилиту top (или даже tl, как описано в разделе *«Постоянное отображение средней загрузки системы в строке заголовка»* **[Совет № 59]**) на каждом сервере приложений при работе с файлом servers.map. Параметр mtime файла будет проверяться при каждом обращении, после чего сервер Apache автоматически перезагрузит карту в случае внесения в нее каких-либо изменений без необходимости перезапуска самого сервера.

Работа с несколькими серверами приложений может оказаться непростой задачей. Очевидно, вам понадобится выполнять синхронизацию файловых систем при помощи NFS или, вероятно, rsync (как описывалось в разделе *«Постоянное отображение средней загрузки системы в строке заголовка»* **[Совет № 59]**). Но иногда работа даже наилучшим образом спланированных схем функционирования серверов может пойти наперекосяк. Что означает появление внутренней ошибки сервера (Internal Server Error) при нажатии кнопки *Обновить браузеры*? Без анализа журнала ошибок на каждом из серверов невозможно определить, какой именно из занятых обслуживанием клиентских запросов, вызвал ошибку серверов. Для решения этой задачи воспользуйтесь следующим правилом перенаправления, позволяющим выбирать, какой именно сервер будет использован для обслуживания данного запроса:

```
RewriteCond %{QUERY_STRING} appserver=(.*)
RewriteRule ^/(.*) http://%1oreillynet.com/$1 [P,L]
```

Теперь можете задавать сервер приложений, который будет обслуживать запросы, добавив в строку URL браузера нужный параметр:

```
http://www.oreillynet.com/some/path/?appserver=www3
```

Таким образом можно быстро выявить проблемный сервер. В сочетании с прокси-серверами для клиентов, циклическим DNS и реплицируемыми базами данных вы получите довольно простую масштабируемую конфигурацию с возможностями расширения до уровня крупной серверной системы.

Дополнительная информация:

- документация по модулю mod_rewrite к серверу Apache, размещенная по адресу *http://httpd.apache.org/docs/mod/mod_rewrite.html*;
- советы, касающиеся прокси-серверов;
- *«Использование rsync в ssh»* **[Совет № 38]**;
- *«Распределение серверной загрузки при помощи циклического DNS»* **[Совет № 79]**;
- *«Настройка репликации в MySQL»* **[Совет № 82]**.

СОВЕТ 100 Обслуживание узлов: массовый веб-хостинг при помощи подстановочных символов, прокси-серверов и перенаправления

(Организация поддержки внутренних веб-серверов)

Предположим, вы работаете с масштабной локальной сетью, скрытой от внешнего мира при помощи маршрутизатора NAT. Структура сети выглядит приблизительно так, как показано на рисунке 8-3.

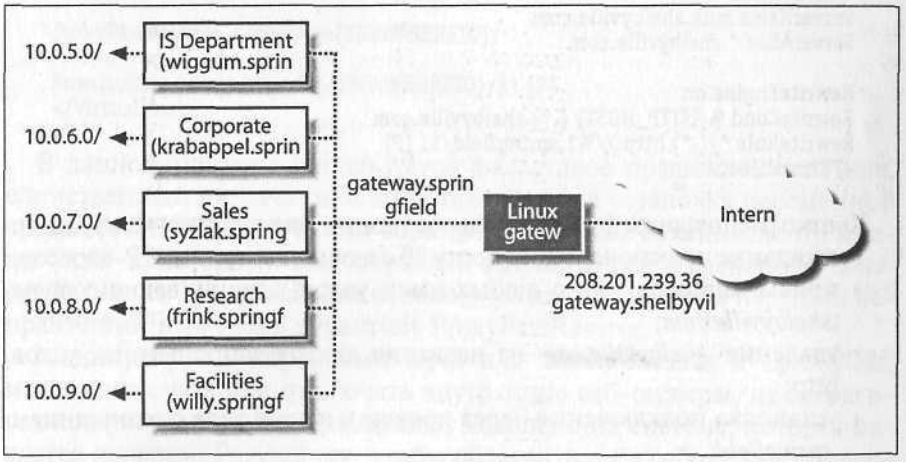

Рис. 8-3. Типичная корпоративная сеть, использующая внутреннюю адресацию и по крайней мере один шлюз, выполняющий трансляцию сетевых адресов для связи с интернетом

Требуется сделать так, чтобы любой компьютер в локальной сети мог быть превращен в веб-сервер. Но, как всякий хороший сетевой администратор, вы достаточно умны и ленивы, чтобы обновлять правила перенаправления на своем брандмауэре каждый раз, когда того требует некоторый клиент. Благодаря аккуратному использованию вышеназванных виртуальных узлов, модулей mod_proxy и mod_rewrite, можете уменьшить свою администраторскую загрузку, ограничив ее

простым обновлением записей DNS. И у вас останется очень мало причин, мешающих полностью «спихнуть» эти обязанности на отделы, захотевшие организовать у себя веб-серверы.

Для начала необходимо запустить сервер Apache с установленными модулями mod_proxy и mod_rewrite на системе, выступающей в качестве шлюза. Кроме того, понадобится DNS-сервер, поддерживающий ваш собственный домен верхнего уровня (см. раздел *«Запуск собственного домена верхнего уровня»* **[Совет № 80]**). Предположим, вы владеете интернет-доменом *shelbyvill.com* и организовали внутренний домен верхнего уровня *.springfield* для обслуживания компьютеров внутри сети.

С этой целью добавьте в файл настройки конфигурации сервера Apache на своем шлюзе следующие строки:

```
Port 80

BindAddress *
NameVirtualHost *

<VirtualHost *>
ServerName mux.shelbyville.com
ServerAlias *.shelbyville.com

RewriteEngine on
RewriteCond %{HTTP_HOST} (.*).shelbyville.com
RewriteRule ^/(.*) http://%1.springfield/$1 [P]
</VirtualHost>
```

Кратко расшифруем функциональное назначение такой конфигурации:

- ожидание подключения по порту 80 с любых доступных IP-адресов;
- прием запросов http с любых имен узлов, заканчивающихся на *.shelbyville.com*;
- удаление *.shelbyville.com* из названия любых запрошенных узлов http;
- установка подключения через прокси к имени узла с окончанием *.springfield*;
- объявление оригинального URI и возвращение результатов клиенту.

Это возможно, поскольку узлы сети в нашем случае задаются по заголовку http в соответствии со спецификацией http 1.1 и не привязываются к конкретному IP-адресу. Сервер Apache будет просматривать содержимое %1 (часть имени перед *.shelbyville.com*) через системный распознаватель и пытаться установить подключение к заданному узлу. Поскольку шлюз использует DNS-сервер, обслуживающий внутренний домен верхнего уровня *.springfield*, он перенаправит запрос на нужный внутренний узел сети.

Предположим, например, что оригинальный запрос был обращен к адресу *http://jimbo.shelbyville.com/index.html*. После обработки в ди-

рективе RewriteCond переменная %1 будет содержать только строку jimbo. Затем сервер попытается установить подключение к узлу %1 (jimbo) с доменным расширением *.springfield*. В результате запрос будет перенаправлен на веб-сервер *jimbo.springfield*, словно никакого шлюза нет.

Это пример простейших настроек, которые, однако, не позволят функционировать веб-серверам, использующим в работе файлы «cookie». Для поддержки этих файлов на серверах, расположенных внутри сети, понадобится несколько модифицировать приведенную выше конфигурацию:

```
<VirtualHost *>
ServerName mux.shelbyville.com
ServerAlias *.shelbyville.com

RewriteEngine on

RewriteCond %{HTTP_HOST} (.*).shelbyville.com
RewriteRule (.*) $1 [E=WHERETO:%1.springfield.com]

ProxyPassReverse / http://%{ENV:WHERETO}/

RewriteRule ^/(.*) http://%{ENV:WHERETO}/$1 [P]
</VirtualHost>
```

В данном примере используется фальшивое правило RewriteRule, единственным назначением которого является установка переменной среды %{WHERETO}. Таким образом, окончание оригинального имени узла, к которому был обращен запрос, будет изменено с *.shelbyville.com* на *.springfield*. Это необходимо для того, чтобы передать исправленное имя узла в директиву ProxyPassReverse.

Манипулируя настройками DNS для *.shelbyville.com* и *.springfield*, можно включать или отключать внутренние веб-серверы, не затрагивая конфигурацию сервера Apache, заданную на системе, которая является шлюзом. Разумеется, чтобы облегчить себе работу можете воспользоваться в записях DNS подстановочными символами для *.shelbyville.com*:

```
*.shelbyville.com IN A 12.34.56.78
```

где вместо 12.34.56.78 необходимо подставить внешний IP-адрес шлюза. Теперь запросы, направленные по адресу *произвольное_имя.shelbyville.com*, поступят сначала на шлюз, а затем на прокси-сервер, без повторного изменения файла зон. В результате вам останется только внутреннее обслуживание DNS (домена *.springfield*). Простейший способ распределения ответственности состоит в делении домена верхнего уровня на несколько вложенных доменов, обслуживание которых ляжет на плечи других внут-

ренних DNS-серверов. Например, можете поместить следующие строки в файл *named.conf*:

```
zone "wiggum.springfield" {
type slave;
file "wiggum.db";
masters { 10.42.5.6; };
};

zone "krabappel.springfield" {
type slave;
file "krabappel.db";
masters { 10.42.6.43; };
};

zone "syzlak.springfield" {
type slave;
file "syzlak.db";
masters { 10.42.7.2; };
};
```

и т.д. Теперь каждый отдел будет иметь свой собственный ведущий DNS-сервер, с помощью которого сможет добавлять новые узлы сети, а также новые веб-серверы, доступные извне, даже без необходимости уведомлять вас об этом по электронной почте. Доступ к новым серверам будет возможен при помощи адресной строки типа *http://ralph.wiggum.shelbyville.com/*, преобразуемой в вид *http://ralph.wiggum.springfield*, причем ее поиск будет осуществляться в записях DNS-сервера отдела, отвечающего за домен wiggum. При работе с сетью интернет информация будет поступать напрямую со шлюза и внешние клиенты даже не узнают о существовании домена *.springfield*.

А что вы будете теперь делать со временем, которое сэкономили благодаря прочтению этой книги?