Безопасность и надёжность системы

Когда речь идёт о безопасности компьютерной системы, надо понимать, что безопасность должна трактоваться с двух сторон:

- безопасность --- надёжность;
- безопасность --- секретность.

Слово *security*, кстати, часто означает оба случая.

С точки зрения пользователя, не важно, отчего пропадут данные: из-за того, что сгорит винчестер или из-за того, что компьютер взломает человек и уничтожит их. И основная задача безопасности --- не ловить злоумышленников, а обеспечивать надежность и секретность хранения и передачи информации. А выявлять взломщиков --- это даже не инструмент решения задачи безопасности, это, скорее, превентивная мера. Поэтому для повышения безопасности надо повышать как секретность, так и надёжность системы.

Инструменты повышения надёжности достаточно очевидны --- данные надо дублировать, повышать качество оборудования и программ и т.д. Инструменты повышения секретности тоже ясны --- надо шифровать всё, что нужно и повышать криптостойкость шифров. А борьба со взломщиками --- третья производная обеспечения безопасности.

Хотя здесь не будет приведена полная классификация уязвимостей, укажем, что проблема безопасности --- троякая:

- может пострадать аппаратное обеспечение;
- могут пострадать данные (в том числе и при работающем аппаратном обеспечении: их может по ошибке стереть пользователь или преднамеренно украсть злоумышленник);
- может произойти отказ в обслуживании: аппаратное обеспечение исправно, данные на месте, но машина работает не так, как мы хотим (например, очень медленно).

Надёжность

Обеспечение надежности в случае компьютерного класса видимо важнее, чем обеспечение секретности. В понятие надежности входит надёжность аппаратного обеспечения (в т.ч. и во враждебных внешних обстоятельствах). Тут от администратора мало что зависит --- если вам доверена покупка аппаратного обеспечения, то необходимо обращаться к доверенным маркам и поставщикам. К сожалению, в образовательных учреждениях это часто невозможно. Тем не менее, если есть возможность выбирать, необходимо выбирать более надёжное железо.

Предположим что, так или иначе, у нас есть аппаратное обеспечение. Однако у любой надежности аппаратного обеспечения есть предел, и мы хотим обеспечить большую надёжность. В первую очередь, нужно определить, а стоит ли овчинка выделки. В компьютерном классе стоит ли повышать надёжность работы конкретного компьютера, если на нём не хранится никаких данных, и он, в конце концов, заменим? Тем не менее, если это необходимо, то типичный метод повышения надежности --- холодный резерв (запасные комплекты аппаратного обеспечения) --- но он, конечно, не спасает от потери данных. Тем не менее, если есть задача обеспечить учебный процесс на N учеников, то было бы неплохо иметь 1-2 компьютера в холодном резерве.

- Достоинства холодного резерва:
    - не нужно предпринимать дополнительных действий.
- Недостатки:
    - не спасает от потери данных;
    - надо предусмотреть оперативное введение резерва в строй.

Второй стандартный способ --- резервное копирование. Если холодный резерв --- это запас железа, то резервное копирование --- запас данных. Если вы хотите обеспечить сохранность данных, то было бы неплохо чтобы эти данные существовали более чем в одном экземпляре и эти экземпляры периодически синхронизировались бы с основным.

Следует понимать, что хранение резервной копии на том же физическом носителе --- достаточно бессмысленная идея, поскольку спасает только от ряда действий пользователя, но не от аппаратного сбоя. Если нужно действительно повысить надежность, то необходимо использовать два диска. В случае, если есть возможность переписывать на другой компьютер, то это ещё лучше, поскольку на компьютере с двумя дисками может неудачно сгореть блоки питания. Если есть возможность передать данные в другое помещение --- лучше в другое помещение. Можно использовать также переносные устройства (прежде всего внешние жесткие диски и флеш-накопители).

В текущих дистрибутивах ПСПО никаких автоматических средств для этого не включено, поскольку сервер не значился в госзаказе.

Одна из схем резервного копирования: на двух машина есть два диска: первая машина пишет свои данные на диск второй, а вторая --- на первую. Такая нехитрая идея позволяет при порче одной машины восстановить данные с другой. При необходимости это можно организовать при помощи программы rsync и службы удаленного терминала SSH, о которой будет подробно рассказываться далее.

Простейший случай использования `rsync` для архивации следующий. Пусть на обоих компьютерах есть пользователь `user`, а на компьютере, куда осуществляется резервное копирование, установлена и запущена служба SSH:

```
# apt-get install openssh-server && service sshd start
```

Тогда на другом компьютере можно дать следующую команду для архивации (в примере 172.16.1.10 --- IP-адрес резервного компьютера):

```
$ rsync -avh /home/user user@172.16.1.10:/home/
```

В результате вся домашняя папка пользователя `/home/user` будет полностью скопирован на другой компьютер. Для ускорения работы утилита `rsync` пропускает файлы, не изменённые со времени последнего копирования. Аналогичным образом утилита `rsync` может копировать данные на внешний носитель:

```
$ rsync -avh /home/user /media/FLASH
```

Есть ещё один метод дублирования информации, менее популярный, чем первые два --- горячий резерв. В этом способе вместо одного устройства работой занимаются сразу два или более устройств. Проще всего это пронаблюдать на примере винчестеров. Для этого существуют несколько способов объединения их в одно пространство. В простейшем случае это зеркалирование. Большинство современных компьютеров делают вид, что они умеют это делать на аппаратном уровне. К сожалению, часто это всего лишь небольшая модификация для BIOS для корректной загрузки с такого объединения, а вся нагрузка ложится на драйвер, то есть программную часть. В Линукс в этом смысле лучше использовать программное объединение дискового пространства в так называемый "том". Интерфейс установщика системы позволяет делать это очень легко --- там разделяется понятие раздела и тома (как набора одинаковых разделов на разных дисках). Поэтому установить ПСПО на машину с двумя винчестерами, сделав из одного них зеркало другого, очень легко. Единственное, что надо сделать --- настроить загрузку.

В случая наличия полноценного аппаратного RAID-конторроллера (это довольно дорогое устройство, по цене сопоставимое с системным блоком) можно установить систему и на

него, если для него есть линуксовый драйвер или он уже поддерживается ядром. При этом установщик может подгрузить все необходимые модули.

Следует помнить, что горячее резервирование дисков не спасает, например, от резкого повышения напряжения или крысы в корпусе, потому что в этом случае выйти из строя могут оба жестких диска. Поэтому при хранении ценной информации зеркалирование дисков должно дополнятся копирование на внешний носитель или удаленный компьютер, находящийся гдето в другом здании, на случай пожара.

Существуют способы горячего резерва и других комплектующих, и даже целых компьютеров, но в школе они редко применимы. Например, одновременно работают две машины, выполняющие одну и ту же функцию, объединенные при помощи специального программного обеспечения так, что при выходе из строя одной машины продолжает работать другая. Аналогичное возможно и с блоками питания, и с процессорами в пределах одной машины.

Следует также сказать, что вопрос обеспечения надежности это задача администратора, и он упирается в большое количество работ, когда вы компьютера вообще не видите --- прокладка кабеля, обеспечение питания, регулярная профилактика. В числе прочего, если важна бесперебойная работа чего-то, то неплохо бы обеспечить источник бесперебойного питания (UPS). Существует ПО для работы с ним, как поставляемое разработчиком, так и отдельное, и можно автоматизировать процесс: если источник питания сообщает что он сможет обеспечить питание только на 15 минут, то машину следует автоматически выключить по прошествии десяти.

Локальная безопасность

Рассмотрим сначала вопросы обеспечения безопасности локальной работы компьютера, не касаясь вопросов безопасной передачи данных по сети или сетевых атак.

Доступность: физическая и виртуальная

Первое, о чем необходимо помнить: борьба за безопасность начинается с физического уровня. Если у злоумышленника есть возможность развинтить компьютер, чтобы получить информацию, то он это сделает. Таким образом, единственный защиты информации при возможности физического доступа к ней --- шифрование диска на уровне его контроллера. Поэтому защита данных на уровне их секретности начинается с административных мер, а не с программных. В первую очередь, никто не должен иметь физический доступ к серверу и физический доступ к его консоли: это позволит злоумышленнику легко войти в систему, даже не зная пароль администратора, например, используя параметр ядра `init=/bin/bash`.

Поднимаясь на уровень выше, рассмотрим уровень программный. В целом, в Unix-подобных системах успешно функционирует система разграничения прав: если права выставлены грамотно, то нет возможности их нарушить, и система достаточно надёжна. Более того, очень многие службы не стартуют, если секретные данные доступны не только пользователю данной службы и администратору.

Для облегчения манипуляций с правами в ПСПО существует утилита `umask`, изменяющая права доступа, которые присваиваются новым файлам и директориям по умолчанию, иными словами, снимающая те или иные права. Например, `umask 022` означает снятие бита записи в группах доступа g и o, то есть права вида 755.

Поскольку ПСПО состоит из различных программ, которые имеют разные уровни доступа к разным объектам системы, необходимо сделать так, чтобы процессы какого-нибудь случайного пользователя системы не навредили нашим объектам. Итак, одним пользователями мы разрешаем запускать те или иные процессы, а другим --- нет.

Атаки отказа в обслуживании

Стоит отметить, что, говоря о надёжности системы, мы говорим в том числе и об атаках на отказ в обслуживании (DoS-атаки, от англ. *Denial of Service*). Хотя под такими атаками сейчас часто подразумевают сетевые атаки (которые более точно называются DDoS-атаками, от англ. *Distributed Denial of Service*)), отказ в обслуживании может быть вызван и действиями одного локального пользователя. Поэтому необходимо сделать так, чтобы в результате некоторых действий локального пользователя система не перестала бы быть способной выполнять полезную работу.

Когда мы говорим о том, что необходимо обеспечивать разграничение прав в системе, нельзя забывать о том, что существуют такие ресурсы, как оперативная память и процессорное время. И то, и другое при определенном стечении обстоятельств может быть загружено в достаточной степени для того, чтобы система перестала работать в нормальном режиме. В Unix-системах нельзя одним процессом, если он не запущен от пользователя root, занять всё процессорное время. Это происходит из-за ограничений приоритета процесса --- его нельзя повысить, если Вы не вошли в систему как root, можно только понизить. Но есть ещё ресурс количества процессов, принадлежащих одному пользователю: если вы хотите занять процессорное время, очевидно, ваша задача --- запустить как можно больше процессов.

Весьма популярным способом DoS-атаки системы с плохо настроенным ограничением на число процессов одного пользователя --- так называемая fork-бомба, то есть программа,

которая не делает ничего, кроме как с максимальной скоростью создает собственные копии в больших количествах, в идеальном случае занимая всю память и большую часть процессорного времени. Для этого потенциальному злоумышленнику достаточно выполнить, например, следующую команду.

```
$ bomb() { bomb | bomb& }; bomb
```

Она создает шелл-функцию с именем bomb, которая начинает рекурсивно запускать саму себя дважды, при этом создается два новых процесса, которые в свою очередь порождают четыре процесса, и так далее. Очевидно, что без ограничений на количество процессов, запущенных одним пользователем, такая команда быстро вызовет отказ обслуживания. Просто посылать сигналы на завершение работы этих программ совершенно бесполезно, поскольку, даже если было достигнуто предельное число процессов, как только один из них завершит работу, на его место тут же встанет новый процесс. В принципе, можно попытаться даже написать собственную fork-бомбу, задача которой --- убивать предыдущую, но, скорее всего, это только усугубит проблему. Вариант выхода из такой ситуации --- послать всем бомбовым процессам не `SIGKILL`, а `SIGSTOP`. Тогда они прекратят своё выполнение, но не будут удалены из таблицы процессов; они будут всего лишь приостановлены, и после остановки их можно будет легко завершить. При возможности управлять ситуацией от лица суперпользователя будет полезно воспользоваться утилитой killall, выполнив команды:

```
# killall -u <имя запускающего такие процессы пользователя> -STOP
# killall -u <имя запускающего такие процессы пользователя> -KILL
```

Однако ясно, что правильным методом борьбы с бомбами является ограничение числа процессов пользователя, что и сделано в современных системам GNU/Linux.

Ограничение ресурсов, выделяемых процессам пользователя

Различные ограничения, накладываемые на пользовательские процессы, задаются при помощи системы `ulimit`. Полную информацию о том, насколько Вас ограничили, можно узнать, выполнив команду `ulimit -a`.

```
$ ulimit -a
core file size          (blocks, -c) 0
data seg size           (kbytes, -d) unlimited
max nice                        (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 1983
max locked memory       (kbytes, -l) 32
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
max rt priority                 (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 256
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Эти параметры записаны в файле `/etc/security/limits.conf`. Ограничивать можно и отдельных пользователей, и группы. Все эти настройки записываются в окружение оболочки входа в систему (*login shell*) пользователя таким образом, что не могут быть изменены, и, соответственно, наследуются всеми процессами пользователя.

Определённым недостатком всей имеющейся у нас системы безопасности является тот факт, что всё, что не запрещено --- разрешено, что даёт некий простор для творчества людей,

которые могут и хотят творить в этой области.

Пароли и их криптографическая стойкость

Речь здесь пойдет о паролях и об аутентификации. Очевидно, что утечка идентификационной информации равнозначна утечке той информации, которая доступна при аутентификации с помощью этого идентификатора. Но иногда есть и другие способы получить доступ к тем или иным образом защищённой информации. Например, в концепции класса активно используется сетевая файловая система NFS, авторизация в которой обычно осуществляется по IP-адресам. Иными словами, любой человек, имеющий пользователя root на локальной машине, теоретически может производить опеределенные (в том числе вредоносные) операции на сервере. Или же, файлы с сильно ограничивающими к ним доступ правами случайно копируются на файловую систему, которая не поддерживает права доступа --- также не очень приятная ситуация.

С довольно давних пор в POSIX-системах нигде на диске не хранятся пароли в явном виде --- только невосстановимые хэши от них, получаемые применением к паролям хэш-функции, которая преобразовывает их в битовую строку фиксированной длины, из которой пароль восстановить нельзя. Хэш отличается двумя важными свойствами --- у одинаковых данных он одинаков, и очень трудно подобрать разные данные, для которых он будет одинаков (и если такой алгоритм обнаруживается, то переходят на более стойкую функцию вычисления хешей). Таким образом, легко сверять пароль по совпадению хэша от пароля.

В отличие от традиционного для большинства систем файла `/etc/shadow`, в котором хранятся хэши паролей всех пользователей, в ПСПО используется система, взятая из TCB (Trusted Computing Base), когда для каждого пользователя имеется свой файл shadow, находящийся в соответствующем подкаталоге `/etc/tcb`. Это позволяет,например, не держать утилиту passwd как suid root.

Для затруднения подбора пароля так называемым "индексированием" хэша --- когда злоумышленник проверяет хэши у различных паролей, и при совпадении хэша он узнаёт правильный пароль --- есть несколько взаимодополняющих путей. Первый --- это усложнение алгоритма вычисления хэша, например, применение функции несколько раз для замеделния процесса вычисления хеша. При однократной проверки при входе в систему это время все равно будет пренебрежимо мало, но при переборе большого числа значений оно становится очень ощутимо. Второй, гораздо более важный --- при первоначальном подсчёте хэша он считается не только от пароля, но и он некоей дополнительной информации, получаемой случайно и именуемой *salt* (соль). В итоге в shadow каждого пользователя сохраняется собственно хэш, соль и количество применений хэш-функции к паролю (обычно менее десяти).

Криптографическая стойкость (она же надёжность) пароля в ПСПО и OWL проверяется при помощи passwdqc-enforce (Password Quality Control), подсистемы, которая позволяет задать некоторые ограничения, которым должен соответствовать пароль, например: минимальная длина при использовании одного, двух, трёх, четырёх классов символов, максимальная длина, состоит ли пароль из словарных слов или нет. Очевидно, что чем больше различных классов символов в пароле, тем он может быть короче --- но тем труднее его будет запомнить. Понятно, что по умолчанию отключено создание паролей из символов одного класса, как бы длинны они не были. Ведь это означает теоретическую возможность создать пароль из, к примеру, большого числа одинаковых символов, который нельзя назвать стойким ко взлому.

В числе прочего не рекомендуется использовать имя пользователя как часть пароля, пароль в одно слово и так далее. Всё это не даст сделать система контроля качества паролей, но он не везде включён, например выключен при выполнении от пользователя root. Суперпользователь будет лишь предупреждён о нестойкости пароля, но изменён он, тем не

менее, будет. Стоит заметить, что это настройки passwdqc-enforce по умолчанию, которые можно изменить в соответствующем объекте подсистемы control.

Рассмотрим пример успешного обновления пароля. Так, passwdqc-enforce предлагает пароль, который, по её мнению, достаточно криптостоек --- произвольные словарные слова, разделённые произвольными знаками препинания:

```
[user@demo ~]$ passwd
Changing password for user.
Enter current password:

You can now choose the new password or passphrase.

A valid password should be a mix of upper and lower case letters,
digits, and other characters.  You can use an 8 character long
password with characters from at least 3 of these 4 classes, or
a 7 character long password containing characters from all the
classes.  An upper case letter that begins the password and a
digit that ends it do not count towards the number of character
classes used.

A passphrase should be of at least 3 words, 12 to 40 characters
long and contain enough different characters.

Alternatively, if noone else can see your terminal now, you can
pick this as your password: "count!diesel!noise".

Enter new password:
Re-type new password:
passwd:
all authentication tokens updated successfully.
```

Часто используемые алгоритмы составления достаточно стойких паролей: строчка из песни, первые буквы чего-нибудь, русские слова в английской раскладке, желательно, чтобы там были не только буквы, но и знаки препинания (русские буквы Э, Ж и тому подобное). Пока это работает, хотя, судя по всему, вскоре программы перебора паролей будут искать и по этим алгоритмам. Подбор пароля по словарю может занять доли секунды. Лектору на данный момент точно неизвестны подборщики паролей, которые перебирают по паролям в стиле leet русским (Н@ТТрumер, ВоТ Т@К). Кроме того, там может присутствовать не только посимвольная замена, в отличие от классического leetspeak (к примеру, }|{, 9|).

Наконец, пароли не надо записывать, их надо запоминать. Существуют программы хранения паролей, которые их защищают другим паролем.

Различные способы узнать ваш пароль

Также не стоит забывать о том, что какой бы пароль криптостойкий не был, всегда существует способ не подобрать его, но тем или иным способом узнать непосредственно. Существует такой класс программ как "клавиатурные воры", или кейлоггеры, которые записывают каждое нажатие на клавиатуру. В системе GNU/Linux у неё есть два основных пути: системная консоль (что потребует прав root и модификации ядра, а при таком уровне доступа можно и более простым путём украсть данные) и X-сервер, а также перехват на уровне ядра операционной системы. Архитектура X-сервера такова, что может существовать программа, которая перехватывает все события с клавиатуры и перенаправляет их конкретным приложениям. Так работают, например, все оконные менеджеры. Поэтому, работая на недоверенной машине, нельзя быть уверенным, что такая программа не включена.

Эмулятор терминала xterm имеет в себе оригинальный способ решения проблемы

кейлоггера: существует режим работы Secure Keyboard, когда сам xterm становится программой, которая захватывает клавиатуру. При включении этого режима xterm инвертирует цвет терминала, и в случае запуска другой захватывающей клавиатуру программы --- а такая может быть только одна --- цвета изменятся обратно, что будет служить знаком того, что ценные данные вводить не стоит. Тем не менее, это не слишком надежный способ контроля перехвата клавиатуры, поскольку в случае недоверенной машины он может быть также произведен на уровне ядра операционной системы.

Следует отметить, что X-сервер и приложение могут находится на разных компьютерах, а для подключения к запущенному X-серверу приложения должны проходить авторизацию. Авторизация бывает двух видов: по хостам (довольно ненадёжно, вдруг какой-нибудь другой пользователь запустит кейлоггер), и по MIT Magic Cookie (специальному хэшу). Первый регулируется утилитой xhost, второй --- xauth. Второй, в принципе, достаточно надёжен, если не принимать во внимание то, что сам по себе сетевой трафик по протоколу X не шифруется. Поэтому, даже если вы уверены, что если вам за спину никто не смотрит, работать через сеть с X-приложениями следует только по SSH (о котором речь пойдет далее), иначе некоторая часть информации может быть просмотрена нежелательными людьми.

Безопасность: сетевая секретность

Проблема открытой передачи пароля

Многочисленные сетевые службы используют идентификацию пользователя по имени (логину) и паролю. Однако для того, чтобы пользователь мог передать свой пароль, охранив его в секрете от третьих лиц, следует убедиться, что он шифруется при передаче по сети и что это шифрование достаточно стойкое. Необходимо достичь того, чтобы передачу никому нельзя было расшифровать --- никому, кроме сетевого сервиса, которому этот пароль и высылается. В противном случае ожидать, что где-то на пути от клиента до службы на одном из компьютеров пароль может быть "подсмотрен".

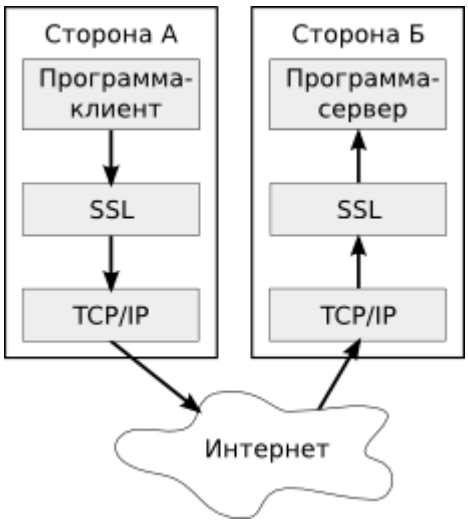
В документации man iptables содержится сводная таблица сервисов, которые не отличаются надежностью в плане безопасности. К ним относятся ftp, telnet, http, ipmap и pop3 --- все они передают идентификационные данные в незашифрованном виде.

От некоторых из этих небезопасных служб следует вообще отказаться, например, от telnet и от ftp с авторизацией пользователя. Есть замечательная утилита dsniff, в документации к которой (`man dsniff`) перечислены протоколы, из которых она способна "выудить" пароли.

Поскольку сами протоколы верхнего уровня надежным шифрованием паролей часто не занимаются, то стоит рассмотреть возможности, позволяющие зашифровать сам трафик. Если внутри протокола не предусмотрено шифрование, то можно попробовать зашифровать передачу --- организовать секретный канал и передавать пароли в чистом виде. Это разумное решение при условии, что канал надёжен. Однако здесь следует напомнить, что транспортный протокол TCP не обеспечивает надежную с точки зрения безопасности доставку: передаваемые данные могут быть как подсмотрены, так и подделаны.

В качестве надежного протокола передачи стоило бы обратить внимание на сетевой протокол IPsec (поверх которого может работать TCP), который замечательно реализован в IPv4, но он не работает "из коробки", и комплекс мер по его включению и настройке достаточно сложный, и если совершить какие-нибудь ошибки, то он либо просто не заработает, либо секретность не будет обеспечена. Поэтому из всех протоколов обеспечения безопасной передачи наиболее важно рассказать про SSL. Протокол SSL (*Secure Socket Layer*) --- это некая прослойка между прикладным и транспортным уровнями сети, обеспечивающее шифровку информации перед ее передачей протоколу TCP и дешифровку информацию, полученной от уровня протокола TCP. Для шифрования данных SSL организует первичный безопасный обмен временными ключами, которые в дальнейшем используются как ключи шифровки и дешифровки. #Для начального обмена

У данного слоя есть выход на прикладной уровень, чтобы приложение знало, что оно использует SSL.



Поскольку SSL не имеет своего набора портов, то разные номера портов TCP применяются для того, чтобы различать обычную службу и такую же службу, использующую SSL. Напрмер, стандартный порт сервера протола HTTP --- 80, а протокола HTTPS (HTTP поверх SSL) --- 443. С помощью утилиты stunnel можно сделать любое "обычное" соединение использующим SSL, в частности, протоколы HTTP, IMAP, POP3, XMPP могут работать поверх него.

Главное достоинство работы через SSL --- прозрачность с точки зрения вышестоящего прикладного протокола. После запуска утилиты stunnel приложение, работающее с каналом, "не заметит разницы" в своих действиях. Главный недостаток SSL --- как только эта схема перестаёт работать (это имеет место быть например в случае протокола FTP), то сразу возникают проблемы: во-первых, клиентское приложение должно знать, что есть специальный нестандартный порт, по которому надо слушать, и, во-вторых, внутри прикладного протокола нет специальных способов управления аутентификацией. За все это отвечает уровень SSL, и приложению придется организовать его самому в отсутствие stunnel.

Кроме протокола SSL, есть близкий способ обеспечения безопасности --- подобный протоколу SSL протокол TLS. TSL, строго говоря, не является протокола, не являющийся отдельным протоколом --- это модификация какого-либо протокола прикладного уровня. Соответственно, на прикладном уровне возможности TLS гораздо шире, чем у SSL. Например, с точки зрения SSL все веб-сайты на одном IP-адресе являются идентичными, поэтому протокол HTTPS может корректным образом защищать только один веб-сайт на одном IP-адресе. Поскольку на уровне SSL никто о разных именах доменов не знает (это адреса протокола HTTP), то у этих хостов один и тот же IP-адрес и ключ SSL. В TLS это вынесено на уровень приложения, то есть уже приложение решает, как шифровать и каким ключом. В отличие от SSL, когда протокол прикладного уровня не знает о его наличии, протоколам, поддерживающим TLS, известно, работает ли он в данный момент или нет.

К примеру, в протоколе IMAP есть расширение, позволяющее включить TLS как команду самого протокола. С точки зрения приложения TLS --- часть прикладного протокола, команда уровня протокола IMAP. Так же дела обстоят и в случае с протоколами POP3, FTP и XMPP. Главный недостаток TLS --- необходимость модификации протокола прикладного уровня, но зато с его помощью можно защитить использование FTP, что невозможно при применении SSL.

Подводя итог, можно сказать, что SSL --- это универсальное туннелирование, а TLS --- метод модификации протокола уровня приложения.

Организация шифрования передаваемых данных

Для шифровки информации в SSL используется симметричное шифрование: для расшифровки используется тот же ключ, что и для шифровки. Для того, чтобы обменяться этим ключом, который является временным, стороны используют асимметричное шифрование при обмене служебной информацией.

При асимметричном шифровании используются два ключа --- открытый и закрытый. С помощью любого из них можно расшифровать то, что зашифровано с помощью другого ключа. Для того, чтобы кто-то мог расшифровать данные, зашифрованные вашим закрытым ключом, необходимо передать ему открытый ключ. В случае успеха расшифровки с использованием закрытого ключа будет знать, что зашифровали их именно вы, если, конечно, ваш закрытый ключ надежно скрыт. Для проверки успешности используется дешифрованная контрольная сумма, которая была приложена к переданным данным. Если дешифрованная контрольная сумма совпадает с контрольной суммой от дешифрованных данных, то расшифровка прошла успешно. Описанный способ называется электронной подписью.

Шифровка открытым ключом так же расшифровывается только с помощью закрытого, при это подобрать к открытому ключу закрытый чрезвычайно трудно. Таким образом, асимметричное шифрование позволяет безопасно передать информацию, используя для шифровки открытый ключ, а в роли этой информации может выступать временный ключ для симметричного шифрования.

Главное достоинство асимметричного шифрования --- безопасность напрямую зависит только от сохранности закрытого ключа, а поскольку этот ключ никак не фигурирует в акте передачи, то похитить его можно только взломав систему, где он хранится. Хотя начальный обмен информацией в SSL устроен более сложно, его идея безопасного обмена основывается именно на применении асимметричном шифрования при начале соединения.

Главный недостаток асимметричного метода --- каждой стороне необходимо обладать подтвержденными открытыми ключами всех других сторон. Если нет уверенности, что это именно их ключи, то возможна следующая ситуация: вы передаёте данные, шифруя их каким-то ключом, но этот ключ принадлежит не вашему адресату, а человеку, подсунувшему его вам на пути к конечной точке. Этот человек в середине (англ. *man-in-the middle*) расшифровывает данные своим закрытым ключом, перешифровывает их с помощью открытого ключа вашего адресата, и передаёт дальше. Такой человек должен перехватывать все пакеты, проходящие между вами. Мест перехвата потенциально много, так как пакеты, например, зачастую проходят через несколько маршрутизаторов на своём пути, и каждый администратор такого маршрутизатора может провести такую атаку.

Есть два варианта решения этой проблемы. Первый --- вы должны как-то действительно убедиться в том, что это открытый ключ вашего адресата. Другой способ состоит в том, что этот открытый ключ попадает к вам не просто так, а подписанный с помощью какого-нибудь из открытых ключей, которые у вас уже есть и которому вы доверяете (ключ "цифрового нотариуса"). На практике используются оба подхода: в случае GPG вы подписываете ключи

людей, которым доверяете, а в случае web-сайтов есть фирмы, бизнес которых состоит в подписывании ключей сервера, использующего HTTPS. Следует отметить, что метод с доверенным нотариусом не дает 100% гарантию, но является достаточно безопасным для некритичных приложений.

Допустим, что стороны решили проблему определения аутентичности открытых ключей своих партнеров, а закрытые ключи надежно хранятся. В этом случае SSL дает довольно высокую гарантию, что трафик защищён, так как закрытый ключ есть только у вашего собеседника. При этом важно чтобы его не было только при первом подключении, так как затем можно запомнить открытый ключ абонента. В случае, если абонент стал предъявлять другой закрытый ключ, то об этом следует немедленно уведомить пользователя.

Управление пользователями

Хранение регистрационной информации

Большая часть регистрационной информации о пользователях хранится в файле `/etc/passwd`. Этот файл доступен на чтение всем пользователям системы, поэтому сведения о паролях (заметим в скобках, что сами пароли вообще не хранятся открытым текстом), времени их жизни и некоторые другие данные из этого файла исключены и находятся либо в файле `/etc/shadow` (в большинстве дистрибутивов ОС Linux), либо в `/etc/tcb/имя_пользователя/shadow` (в дистрибутивах ПСПО и некоторых других). И в том, и в другом случае доступ к этой информации существенно ограничен с помощью механизма разграничения прав доступа в Unix-подобных ОС.

Рассмотрим файл `passwd`. Каждая строка в нем соответствует одному регистрационному имени и состоит из полей, разделенных двоеточиями:

```
$ head /etc/passwd
root:x:0:0:System Administrator:/root:/bin/bash
bin:x:1:1:bin:/:/dev/null
daemon:x:2:2:daemon:/:/dev/null
adm:x:3:4:adm:/var/adm:/dev/null
lp:x:4:7:lp:/var/spool/lpd:/dev/null
mail:x:8:12:mail:/var/spool/mail:/dev/null
news:x:9:13:news:/var/spool/news:/dev/null
uucp:x:10:14:uucp:/var/spool/uucp:/dev/null
games:x:12:100:games:/usr/games:/dev/null
ftp:x:14:50:FTP User:/var/ftp:/dev/null
```

Опишем в двух словах, что означает каждое поле.

1. Первое поле в каждой строке содержит регистрационное имя пользователя.
2. Второе поле ранее содержало хэш пароля, однако сейчас в подавляющем большинстве случаев содержит символ `x`, означающий, что соответствующая информация хранится в другом месте.
3. Третье поле --- это идентификатор пользователя (User ID, UID).
4. Четвертое поле --- идентификатор основной (первичной) группы пользователя. Пользователь, разумеется, может входить и в другие группы, однако сведения об этом хранятся не здесь, а в файле `/etc/group`.
5. Пятое поле по историческим причинам носит имя GECOS и содержит так называемое "полное имя" пользователя. Давным-давно создателям UNIX Кену Томпсону и Деннису Ритчи нужно было хранить информацию о том, какой пароль нужно использовать при регистрации в системе, к которой был подключен принтер. Управлялся доступ к принтеру операционной системой GECOS, поэтому именно такое название получило соответствующее поле в `/etc/passwd`. В наше же время в этом поле чаще всего хранится разнообразная дополнительная информация о пользователе: полное имя, номер комнаты, телефон и, возможно, что-либо еще. Соответствующие "подполя" при использовании такой структуры разделяются запятыми. Отметим, что такая возможность, однако, используется не слишком часто, и обычно в это поле заносят только настоящее имя пользователя.
6. Шестое поле содержит имя домашнего каталога пользователя.
7. Последнее, седьмое поле содержит имя программы, запускаемой в качестве пользовательской оболочки (Shell).

Сделаем следующее важное замечание. По значению седьмого поля в `/etc/passwd`

зарегистрированных в системе учетные записи пользователей можно разделить на три группы.

- Если в качестве оболочки указан один из интерпретаторов командной строки, перечисленных в файле `/etc/shells`, то пользователь считается "настоящим": он может "войти" в систему и полноценно ее использовать.
- Если в качестве оболочки указано специальное значение `/sbin/nologin` (иногда --- другое), то пользователь не сможет зарегистрироваться в системе обычным способом (с помощью программы `login`). Тем не менее, если это значение указать в списке допустимых оболочек (`/etc/shells`), то все подсистемы будут считать, что соответствующий пользователь --- "настоящий", а потому разрешат, например, подключение по SSH без запуска оболочки.
- Если же в качестве оболочки указано имя несуществующей программы или, скажем, `/dev/null`, то зарегистрироваться в системе (с помощью программы `login`) пользователь не сможет вовсе. Возможность заведения в системе таких пользователей позволяет контролировать разделение прав доступа к разным частям системы. Дело в том, что при работе различных служб разумно ограничить использование учетной записи суперпользователя, а потому для каждой такой службы заводится специальная учетная запись. С одной стороны, "по-настоящему" зарегистрироваться в системе с ее помощью невозможно (нет пароля и корректной пользовательской оболочки), с другой --- в случае атаки на эту службу под угрозой окажутся лишь ее собственные файлы. Соответствующих таким учетным записям пользователей обычно называют "псевдопользователями", или "системными пользователями". Псевдопользователей в системе бывает довольно много, обычно --- более 40 (это число зависит от набора установленных служб).

Посмотрим теперь в файл `/etc/group`. В нем содержится список зарегистрированных в системе групп пользователей. Каждый пользователь может состоять в нескольких группах, а каждая группа --- содержать несколько пользователей:

```
$ head /etc/group
root:x:0:
bin:x:1:root
daemon:x:2:root
sys:x:3:root,bin,adm
adm:x:4:root
tty:x:5:
disk:x:6:root
lp:x:7:
mem:x:8:
kmem:x:9:
```

Сведения о членстве пользователей в группах оформляются следующим образом:

1. Первое поле --- имя группы.
2. Второе --- зашифрованный пароль группы (практически никогда не используется; `x`, как и для соответствующего поля в `passwd`, означает, что сведения находятся в другом месте).
3. Третье поле --- числовой идентификатор группы.
4. Четвертое поле --- имена пользователей --- членов группы (через запятую).

Опишем теперь, как используется информация о членстве в группах. Каждый раз, когда от имени пользователя запускается тот или иной процесс, автоматически выясняется его "первичная" группа (обычно она соответствует полученному при входе в систему GID), а членство в процесса в остальных группах определяется с помощью просмотра файла `/etc/`

group. Этот файл просматривается в поисках регистрационного имени (не UID!) пользователя --- подходящие строки и определяют результат. Заметим, что можно создать в системе учетные записи разных пользователей с одинаковым UID и организовать для них членство в разных наборах групп. Пользоваться этой возможностью, однако, следует с большой осторожностью.

Управление учетными записями

Создадим нового пользователя. Зададим ему регистрационное имя admin и "описание" Newly Added Admin (оно будет занесено в поле GECOS):

```
# usearadd -G wheel -c "Newly Added Admin" admin
```

Ключ -G отвечает за список "дополнительных" групп, в которые создаваемый пользователь вступает (в данном случае это группа wheel). Поскольку мы не указали ключ -g, то была создана также основная группа для нашего пользователя. Ее имя совпадает с именем пользователя (admin), а идентификатор выбирается первый из доступных (свободных). Точно так же, поскольку мы не указали ключ -u, то и идентификатор пользователя выбирается первый из доступных. Отметим, что нельзя предполагать, что идентификаторы создаваемых пользователя и группы обязательно совпадут.

Сделаем еще одно важное замечание. В современных дистрибутивах ОС Linux принято создавать для каждого "настоящего" пользователя отдельную группу, а все группы с более чем одним пользователем считать "дополнительными". Это даёт некоторые преимущества всмысле безопасности и возможность по числу строк в /etc/passwd оценивать количество пользователей.

Посмотрим, как при добавлении пользователя изменились файлы /etc/passwd и /etc/group:

```
$ grep admin /etc/passwd /etc/group
/etc/passwd:admin:x:501:501:Newly Added Admin:/home/admin:/bin/bash
/etc/group:wheel:x:10:root,user,admin
/etc/group:ftpadmin:x:51:
/etc/group:cvsadmin:x:53:
/etc/group:netadmin:x:104:
/etc/group:wineadmin:x:57:
/etc/group:admin:x:501:
```

Как мы видим, в /etc/passwd появилась новая запись для нашего пользователя (UID = 501), а в /etc/group была создана новая группа (GID = 501). Кроме того, в соответствии с нашим указанием пользователь admin был добавлен в группу wheel. Отметим, что при заведении пользователя был также создан его домашний каталог, куда сразу же скопировались файлы из каталога /etc/skel. Данный каталог обыкновенно содержит минимальный, "базовый" набор конфигурационных файлов для создания комфортного пользовательского окружения и, естественно, может быть модифицирован администратором системы.

Заметим, что применение утилиты grep для выяснения членства пользователя в группах может дать не вполне "чистый" результат: к примеру, имя запрашиваемого пользователя может, как в нашем случае, содержаться в качестве подстроки в других именах. Чтобы избежать таких проблем, правильнее всего воспользоваться утилитой id:

```
$ id admin
uid=501(admin) gid=501(admin) groups=501(admin),10(wheel)
```

Уделим теперь внимание проблеме удаления пользователей из системы. За выполнение этого

действия отвечает утилита `userdel`, манипулирующая перечисленными выше файлами. У этой утилиты есть ключ `-r`, соответствующий удалению не только учетной записи пользователя, но и его домашнего каталога и почты (обычно лежащей в каталоге `/var/mail`). Однако всегда ли оправдано использование `userdel`? Работа этой утилиты приводит к освобождению идентификатора пользователя (UID), который впоследствии может быть присвоен другому, вновь созданному пользователю. Это приведет к тому, что все не удаленные из системы пользовательские данные (к примеру, содержащиеся в сохранившемся на диске домашнем каталоге) окажутся доступны новому пользователю. Понятно, что такой результат, вообще говоря, не может считаться удовлетворительным.

Как же стоит поступить в такой ситуации? Одним из решений может стать запрет на вход (логин) вместо удаления. За такое действие отвечает команда `passwd(8)` с ключом `-l` (`--lock`) (то же самое можно сделать при помощи `usermod -L`). Стоит, однако, обратить внимание, что в такой ситуации не исключается возможность входа по SSH (например, если в конфигурационном файле `sshd_config` отключена опция `UseLogin`). Разумеется, заблокированного таким способом пользователя можно впоследствии разблокировать.

Естественно, учетные данные зарегистрированного в системе пользователя могут быть модифицированы системным администратором. Некоторые из таких модификаций производятся еще не встречавшейся нам утилитой `usermod`. Ее ключ `-e`, к примеру, позволяет указать дату "отключения" учетной записи. Есть также утилита `chage`, модифицирующая информацию о сроках "истекания" пароля, --- с ее помощью администратор может, допустим, установить политику смены пользователями своих паролей с той или иной периодичностью.

Управление паролями

Скажем несколько слов и о пользовательских паролях. При заведении пользователей с помощью `useradd` пароли не создаются, поэтому, чтобы обеспечить возможность входа в систему, нужно создать (вообще говоря, сменить) пароль. Проще всего это сделать с помощью команды `passwd`. Заметим, что в дистрибутивах ПСПО существуют две утилиты с таким именем: пользовательская из `/usr/bin` и администраторская из `/usr/sbin`. Первая позволяет любому пользователю системы сменить пароль на свою учетную запись, а второй мы сейчас воспользуемся для задания пароля созданному нами пользователю `admin`.

Отметим, что пароль на вход команды `passwd` можно задать разными способами. Первый способ --- задать пароль интерактивно (с клавиатуры, по запросу самой утилиты), второй --- воспользоваться ключом `-p` и задать в командной строке хэш пароля. Остановимся на втором способе чуть подробнее.

Чтобы получить возможность получить BlowFish-хэш (значение именно такой функции используется для хранения паролей в дистрибутивах ПСПО), можно написать специальную программу на языке C, использующую функцию стандартной библиотеки `crypt(3)`, а можно воспользоваться средствами Python. Пойдем по второму пути и установим пакет с модулем `bcrypt`:

```
# apt-get install python-module-bcrypt
```

Теперь можно воспользовать двумя нужными нам функциями из этого модуля. Первая из них --- `gensalt`, генерирующая salt для пароля, вторая --- `hashpw`, возвращающая хэш по заданным паролю и salt'y. Простейший вариант совместного использования этих функций выглядит так:

```
$ python -c 'import bcrypt; print bcrypt.hashpw("mypassword", bcrypt.gensalt(8))'
$2a$08$FxyUan62aD2P9rWSJzEcdO1JApy8m/YHYDaHbgCiKVZAykI/XdoOC
```

Именно полученный таким способом хэш и задается в качестве параметра командной строки утилиты `passwd` (после ключа `-p`).

Коснемся еще одной важной темы --- автоматического задания паролей. Если требуется в автоматическом режиме создать большое количество учетных записей с паролями, то можно действовать по-разному. В случае, когда пароль для всех учетных записей один и тот же (или используется заранее известное множество паролей), имеет смысл просто "подсмотреть" хэш и подставить его в нужное место соответствующего сценария. Более изящный способ --- вычисление паролей на основании, допустим, имени пользователя и еще какой-либо информации (пусть этим занимается любая подходящая функция с "секретным" параметром). Можно также генерировать случайные пароли с помощью предназначенной для этого утилиты `pwgen`. В последних двух вариантах значения хэша для паролей можно получать, например, описанным выше способом.

Покажем, как работает упомянутая нами утилита `pwgen`. Вначале установим соответствующий пакет:

```
# apt-get install pwgen
```

А теперь просто дадим команду `pwgen`:

```
$ pwgen
oi1Sieg1 Coovah3o chi1Et7t zei0Ub0A aeQua4Sh Oyei6aij aebe3Eod uafai1Pa
vu1AiFiu Phe4Eep5 aZiesh4o pheeBi2a Ahvo5cu8 Jei9Efae giK2aide Nohf6bah
Gi2heira eis7te6Z rai4eeSh GeeK9ui5 aesh6Ud6 nuiCe5im aphai8oX oChoo9Sh
UChek0He akei6Ien Reib8ahk xooJ3sai vaeL7miu up4saeSo wiith9Es choD7iex
pheV2ohf ahY0ooCh uekohv9A ood8Awae ekaiHah9 Chif5toH iF4ga1es ePhah0Ai
jaiShee7 Asa6GeeB iezou1Do bie7eeRe Luxieng3 keiZee5A Pach3Voo Iish5Wie
aiv5ra5O Quahgo8s deeXae2f ohVa7OhD pieh8aeH HahGhae6 eebai6Je iZooch9I
Bub7pee1 ep9NaPhe Bahveiv5 Eechee8g Wae6jogh Uch8eoxe Mokaet2t ech7OeKa
Aingeew8 zub1ohSo koTae2ek dohn1Ahk Ceim8Foo cee9eeTh UGho4nah Cho4aiTh
shiwee8I faecie0S aKaa8uen iezei6Qu EeHiiha7 Eexiech6 muWea6du ReeYi2he
vei7aiCe aiDohj0o iuSaum3f ca2queeL quaek1Ca ai5Iu3aC ieWe5We6 Cohk3ei9
mo1SeiG8 XeeN7aej GooTh0fa tie5TaeJ bah9ooSh nah2Dai7 aiCe0oeW Bah8Zuye
yohv5Eet aegie5Uz iZie6ca6 iw4Jaib4 oGhee9ie taexoe5A eeY6iehu ibaew6Vu
xie2Ieh8 eihe6eTa ooMe0oga ahli6eB7 ba2eaRah om7eiT2i Ie8shiWe Uiya3eew
Kai9ni2d Shoh2ePh EiZ4fe9Z ieSeipo0 ahxoi5Ee uNochae9 JooXo5uu BareiV0x
aa4Ishi2 jae9ohXa AeFo7Tha Zohthoo1 Ua4eeTho zo9Oa0Ph caiN7zeb bueZ2the
eelui6Ue joo8Baat phee8Tae ieQue4th eiya7eiJ ahGo7tho usoh7AhK mee9Fiec
au7Oow6u ohn8ohJu phij8Toh aigi1EiZ oZahmah0 quooGae3 Quaej7tu Ceexu7da
Quae4huz hoPhei6s Ahp3oopu aeGh7heg sah7ahMa ahCh0iit eiquae2A eixae7Uc
ingahCh5 Yai7ohve ohC3quee joth3Bay voyai8Bo shuNgoo3 wai5Kaiv ieJah3Ei
```

Как видно, эта утилита генерирует случайные пароли. По умолчанию они "человеко-читаемые", а не абсолютно случайные, однако можно потребовать и обратного. Разумеется, с помощью параметров командной строки можно регулировать и длину, и количество генерируемых паролей, а также использовать другие многочисленные возможности этой утилиты.

Управление группами

Отметим, что при создании нашего пользователя `admin` мы выбрали для него лишь одну дополнительную группу --- `wheel`. Было бы полезным, однако, добавить его и в другие группы. Зачем это может понадобиться? Дело в том, что при работе, например, с пользовательским десктопом, полезно выдавать первому "вошедшему" доступ к таким ресурсам, как съемные диски, аудиоустройства и пр. Такие действия обычно производятся специальным модулем PAM, меняющим права к соответствующему набору файлов из каталога `/dev`. В нашем случае таких файлов оказалось довольно много (более двух

десятков):

```
$ ls -l /dev/ | grep user
crw-rw---- 1 user  audio   14,   12 Jul 31  2008 adsp
crw-rw---- 1 user  audio   14,    4 Jul 31  2008 audio
crw------- 1 user  root     5,    1 Jul 31 14:07 console
crw-rw---- 1 user  audio   14,   25 Jul 31  2008 dmmidi1
crw-rw---- 1 user  audio   14,    3 Jul 31  2008 dsp
crw------- 1 user  root    29,    0 Jul 31  2008 fb0
brw-rw---- 1 user  floppy   2,    0 Jul 31  2008 fd0
brw-rw---- 1 user  floppy   2,   84 Jul 31  2008 fd0u1040
brw-rw---- 1 user  floppy   2,   88 Jul 31  2008 fd0u1120
brw-rw---- 1 user  floppy   2,   28 Jul 31  2008 fd0u1440
brw-rw---- 1 user  floppy   2,  124 Jul 31  2008 fd0u1600
brw-rw---- 1 user  floppy   2,   44 Jul 31  2008 fd0u1680
brw-rw---- 1 user  floppy   2,   60 Jul 31  2008 fd0u1722
brw-rw---- 1 user  floppy   2,   76 Jul 31  2008 fd0u1743
brw-rw---- 1 user  floppy   2,   96 Jul 31  2008 fd0u1760
brw-rw---- 1 user  floppy   2,  116 Jul 31  2008 fd0u1840
brw-rw---- 1 user  floppy   2,  100 Jul 31  2008 fd0u1920
brw-rw---- 1 user  floppy   2,   12 Jul 31  2008 fd0u360
brw-rw---- 1 user  floppy   2,   16 Jul 31  2008 fd0u720
brw-rw---- 1 user  floppy   2,  120 Jul 31  2008 fd0u800
brw-rw---- 1 user  floppy   2,   52 Jul 31  2008 fd0u820
brw-rw---- 1 user  floppy   2,   68 Jul 31  2008 fd0u830
srwx------ 1 user  root           0 Jul 31 14:07 gpmctl
brw-r----- 1 user  cdrom   22,    0 Jul 31  2008 hdc
crw-rw---- 1 user  audio   14,   18 Jul 31  2008 midi1
crw-rw---- 1 user  audio   14,    0 Jul 31  2008 mixer
crw-rw---- 1 user  audio   14,   16 Jul 31  2008 mixer1
crw-rw---- 1 user  audio   14,    1 Jul 31  2008 sequencer
crw-rw---- 1 user  audio   14,    8 Jul 31  2008 sequencer2
```

Какое отношение это имеет к группам? Оказывается, смена UID файлам устройств удобна далеко не всегда. Во многих случаях разумнее передавать права на такие файлы не одному пользователю, а целой группе, что дает возможность настроить права доступа "заранее". В том же `/dev`, как несложно заметить, есть большое количество файлов, принадлежащим не группе `root` или `wheel`, а группе `audio`, `floppy` и пр. Для того, чтобы пользователи имели доступ к этим файлам, и нужно включать их в соответствующие группы:

```
$ id user
uid=500(user)                                                    gid=500(user)
groups=500(user),10(wheel),14(uucp),19(proc),22(cdrom),71(floppy),80(cdwriter),8
1(audio),83(radio),119(xgrp),422(camera),423(scanner)
```

Как же добавить пользователя в ту или иную группу? В стандарте POSIX добавление пользователя в группу (равно как и удаление его оттуда) не реализовывалось какой-либо конкретной утилитой. Предполагалось, что соответствующие действия будут производиться системным администратором путем непосредственного редактирования файла `/etc/group`. Была (и, разумеется, никуда не исчезла) также возможность использования утилиты `usermod` с ключом `-G`, который требует явного указания полного списка групп, в которых должен состоять пользователь. Понятно, однако, что если пользователь уже являлся членом десяти групп, то для добавления его в одиннадцатую вначале нужно было получить список первых десяти и только потом добавить к ним одиннадцатую. Такая операция, очевидно, не атомарна, а потому в дистрибутивах ПСПО предусмотрена команда `gpasswd`:

```
# gpasswd -a admin proc
Adding user admin to group proc
```

Как видим, ее использование позволяет избежать описанных проблем. Проверим результат:

```
$ id admin
uid=501(admin) gid=501(admin) groups=501(admin),10(wheel),19(proc)
```

Использование su и sudo

Иногда системному администратору необходимо совершать действия от лица суперпользователя. Заходить под учётной записью root в программе login не рекомендуется, обычно этот метод используется в крайних случаях, когда другие варианты невозможны. Предпочтительней получать привилегии суперпользователя во время сеанса работы из под учётной записи обычного пользователя. На системном уровне подобная схема реализуется с помощью механизма setuid. Команд, способных предоставить обычному пользователю права суперпользователя, две --- su и sudo.

su

Команда `su` (switch user) позволяет получить права любого пользователя, при условии, что вам известен пароль этого пользователя.

Существует 4 режима работы `su`:

- restricted --- выполнять `su` имеет право только root;
- wheel --- выполнять `su` имеют право только пользователи, входящие в группу wheel;
- wheelonly --- использовать `su` могут все, но лишь пользователи, входящие в группу wheel, могут получать с её помощью права суперпользователя;
- public --- ограничения отсутствуют, любой пользователь может с использованием su получить права любого другого пользователя.

В дистрибутивах ПСПО команда `su`, выполненная без параметров, дает права суперпользователя, не меняя при этом окружения. Чтобы изменить окружение, то есть запустить login shell, необходимо использовать `su -`.

Login shell

Когда пользователь начинает сеанс работы с системой, в первую очередь запускается командный интерпетатор, указанный в `/etc/passwd`. Если для входа в систему и работы используется графический интерфейс в качестве основного, то кроме запуска самого командного интерпретатора, в нем выполняются скрипты инициализации графической сессии. Процесс этого командного интерпретатора отличается от процессов других шеллов, запущенных пользователем. Этот процесс называют login shell. В списке процессов к его имени в начале приписывается `-`. Login shell выполняет определённые дополнительные действия, например, устанавливает переменные окружения UID и GID.

sudo

Команда `sudo` обладает более широкими возможностями, чем `su`. Она позволяет уточнять, какие категории пользователей могут пользоваться учётными записями каких пользователей, а также ограничивать список команд, которые могут исполняться таким образом. Например, можно использовать `sudo` для разрешения монтирования cdrom некоторому пользователю.

По умолчанию `sudo` сохраняет окружение пользователя, но это поведение можно изменить (таким образом можно реализовать запуск X-приложений от учётной записи root).

/etc/sudoers

/etc/sudoers --- конфигурационный файл для `sudo`. Редактируется этот файл командой `visudo`. Будучи запущенной без каких-либо предварительных действий, эта команда

откроет `/etc/sudoers` в редакторе `vi`. Для изменения используемого редактора необходимо указать желаемый редактор в переменной окружения EDITOR.

В этом файле задаются пользователи, группы пользователей, команды, которые они могут выполнять, и прочие ограничения. Синтаксис `/etc/sudoers` подробно описан при помощи EBNF в man sudoers.

Рассмотрим простой, однако часто встречающийся на практике пример --- как дать возможность пользователям, входящим в некоторую группу (например, администраторам), выполнять произвольные команды, указывая при этом только собственный пароль:

```
%admin ALL=(ALL) ALL
```

Символ `%` означает, что admin --- название группы, а не имя пользователя.