Пользователи, группы и ещё немного о файлах

В этом модуле мы продолжим знакомство с пользователями.

Типы пользователей и права доступа

Итак, у нас есть пользователь — человек, которым мы логинимся в систему.
Есть суперпользователь root, который обладает неограниченными правами. Мы обращаемся к его правам только время от времени и не логинимся им напрямую. Вместо этого мы используем *sudo*.

И у нас есть некоторые пользователи, которые создала сама операционная система, они нужны системе для выполнения служебных процессов.

Почти всё взаимодействие с Linux и работающими в нём сервисами происходит через текстовые файлы. Пользователи тоже хранятся в текстовых файлах.
Каждый пользователь при создании записывается в файл /etc/password.

Вот как выглядит часть этого файла:

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
```

Технически вам ничего не мешает поправить какие-то свойства пользователя прямо в этом файле, но делать этого не рекомендуется. Всегда можно опечататься и получить проблемы. Для того чтобы дописывать и редактировать этот файл, используются команды, о которых мы поговорим далее.

В нём используются следующие колонки:

1. **Имя пользователя** — то имя, под которым вы логинитесь в операционную систему. Оно используется для идентификации пользователя человеком.

2. **Пароль** — тут вместо него символ x. Это связано с безопасностью, чуть позже мы поговорим об этом подробнее.
3. **Числовой идентификатор пользователя** — UID, операционная система оперирует именно этим параметром для идентификации пользователя. В рамках сервера он уникальный. Операционной системе не очень интересно имя пользователя, так что пользователь root может называться и по-другому, важен только UID. Только у пользователя с UID 0 неограниченные права.
4. **Числовой идентификатор группы** — GID, к какой группе принадлежит пользователь.
5. **Полное имя пользователя и некая дополнительная информация** (например, номер кабинета, где сидит сотрудник, и телефон или краткое описание программы, если это не человеческий пользователь).
6. **Домашний каталог пользователя**.
7. **Шелл пользователя**.

Что интересного можно увидеть в этом списке? Во-первых, странные вещи вроде uucp, news, irc. Это наследие очень давних времён. Интернет принёс новые протоколы и новые инструменты для того, чтобы обмениваться информацией, и старые сервисы медленно отмерли. Эти пользователи остались ради обратной совместимости.

А во-вторых, в этом списке есть очень важные группы. Например, *sudo*.
Только пользователи в этой группе могут воспользоваться командой *sudo*.

Shell — это оболочка командной строки. Это программа, которая предоставляет нам тот самый интерфейс для общения с системой. У нас есть специальный язык для общения с ней, который мы уже начали изучать и использовать. Она принимает от нас команды и отвечает нам на них. Они бывают разными, самые распространённые — это bash, sh и zsh. Чаще всего мы будем говорить о bash или об sh.

Например, у пользователя на примере выставлен bash. У вашего пользователя, скорее всего, также будет выставлен bash.

Обратите внимание, что у некоторых пользователей вместо /bin/bash стоит /usr/sbin/nologin. Это фактически означает запрет на получение интерактивного шелла, такого, как видим мы с вами, когда открываем консоль. То есть у него не будет возможности выполнять произвольные команды.

Это очень полезно с точки зрения безопасности, потому что в случае компрометации учётной записи, злоумышленник ограничится доступом только к одному из сервисов, работающих на сервере. Он не сможет получить доступ к файлам, он не сможет выполнить каких-то действий, которые помогут получить ему больше прав в системе.

В процессе работы вам придётся сталкиваться с различными программами, которые работают от своих пользователей. И может понадобиться этих пользователей создавать.
Кроме того, обычно у каждого инженера эксплуатации есть свой аккаунт на сервере, поэтому и для них создаются пользователи.

Создание и управление пользователями

Для создания пользователей используются команды *useradd* и *adduser*.
Useradd — это часть операционной системы, она обладает очень базовой функциональностью, то есть просто создаст пользователя без дополнительных атрибутов (например, она по умолчанию не создаст для пользователя домашнюю директорию, не позволит добавить комментарий). Она не интерактивная, то есть все атрибуты пользователя придётся задавать прямо в команде. Для удобства управления пользователями, для создания пользователя со всеми необходимыми атрибутами написали программу *adduser* — это гораздо более мощная обвязка над useradd, будем использовать её.

```
sudo adduser victor
```

Для редактирования атрибутов пользователя используется команда *usermod*. В отличие от *adduser* она не интерактивная, то есть нам придётся сразу передать ей все параметры в строке запуска.

```
sudo usermod
```

```
sudo usermod -d -m /home/evil victor
```

Вы могли заметить, что мы говорили: во втором поле файла /etc/passwd должен быть пароль. Но там только буква x. Дело в том, что в определённый момент пароль был оттуда убран и перенесён в файл shadow.

Файл passwd доступен на чтение для всех. Это логично, потому что доступ к нему требуется для работы множеству утилит. Запретить доступ к этому файлу означало бы множество проблем. Хранить там пароль — большой риск для безопасности.

Поэтому пароль и настройки (частота смены и другие) были перенесены в отдельный файл, доступ на чтение к которому есть только у root.

Заглянем в файл shadow.

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

Значимыми для нас являются первые два поля, остальные отвечают за служебную информацию.

Пароль
Значение во втором поле и есть пароль. Но хранить пароль в открытом виде, даже в файле, который доступен только root, всё ещё небезопасно. Как минимум тогда бы администратор системы знал все пароли, а это было бы крайне неэтично.

Поэтому для хранения паролей на аутентифицирующей стороне используется не сам пароль, а его хеш.

Хеш-функция — это функция, которая позволяет из входных данных произвольной длины получить строку фиксированной длины. И если в данные будет внесено любое малейшее изменение, хеш тоже изменится. А из одних и тех же данных будет всегда получаться один и тот же хеш.

Когда пользователь вводит пароль, вычисляется его хеш. И сравнивается с тем, что хранится в файле /etc/shadow. Если они совпадают — был введён правильный пароль. Таким образом, когда мы вводим пароль, операционная система вычисляет значение хеш-функции и сравнивает её с тем, что записано в файле /etc/shadow.

Для смены пароля используется утилита *passwd*.
Пароль лучше периодически менять. Если вы забыли свой пароль, его может поменять для любого пользователя root. Тогда программа не спросит тот пароль, который задан сейчас, а сразу предложит создать новый.

Рассмотрим, какие права доступа на файле, который запускается, когда мы говорим passwd. Чтобы это выяснить, спросим буквально «где passwd»:

```
whereis passwd
```

Исполняемый файл тут только один, так что посмотрим, какие на него выставлены права.

```
ls -la /usr/bin/passwd
```

```
-rwsr-xr-x
```

Буква s означает, что, помимо обычных прав доступа, файлу присвоен так называемый SUID (set user ID). Это механизм, родственный правам доступа, он называется флагами доступа.

Дополнительные возможности

Флаги — это небольшой набор специальных видов прав, которые мы можем применять к файлу.
Выставленный на файл SUID означает, что, когда файл запускается, он запускается с правами владельца файла, то есть в нашем случае суперпользователя, вне зависимости от того, какой пользователь его запустил. Таким образом, программа запущена от нашего пользователя, но с правами рут и может изменить содержимое файла /etc/shadow.

Чтобы выставить такой флаг, тоже используется команда *chmod*:

```
chmod u+s yourfile
```

По аналогии почти такой же эффект на файлы имеет выставленный SGID (set group ID). SGID можно выставить на файл, и он будет запущен с правами группы владельца файла. Но такой случай используется редко.

SGID чаще всего используется для каталогов. Если выставить его на каталог:

```
chmod g+s dir
```

то всеми созданными в нём файлами (вне зависимости от того, кто их создал) будет владеть группа, владеющая каталогом.

Давайте посмотрим на директорию /tmp, в ней хранятся разнообразные временные файлы, которые существуют, пока работает программа, которая их создала. Например, когда запускается графическое окружение, оно создаёт файлы.
Можно предположить, что создавать и удалять файлы и каталоги в ней должны уметь все, ведь программы запускаются под разными пользователями.
Давайте проверим.

```
ls-la /tmp
```

```
Drwxrwxrwxt
```

Да, всё так. Все могут выполнять все действия, и есть «что-то ещё» — буква t?

Это обозначение так называемого sticky bit — липкого бита. Он говорит нам о том, что удалять в директории файлы могут только те, кто является их владельцем.
Для файла стики бит не имеет смысла.

Чтобы выставить стики бит и сделать такую же директорию, в которой удалить файл можно, только если ты же его и создал, используется команда:

```
chmod +t dir
```

Группы пользователей

Группы нам с вами нужны для объединения пользователей по одному признаку. Например, несколько программ, которые работают с одними и теми же файлами, могут запускаться от разных пользователей, но в одной группе.

Файлы для управления группами очень похожи на аналогичные для управления пользователями /etc/groups и /etc/gshadow.

Файл /etc/groups очень простой, в нём есть поля имени группы, состава группы, идентификатора и поле под пароль.

Для создания группы используется команда *groupadd* с именем группы.

Давайте добавим группу devops:

```
sudo groupadd devops
```

Посмотрим на неё в файле /etc/group:

```
cat /etc/group
```

Теперь попробуем добавить в неё пользователей:

```
cat /etc/passwd
```

Теперь добавим нашего пользователя в группу, но так, чтобы он остался членом своей основной группы:

```
sudo usermod -G -a devops victoria
```

А вот нового пользователя создадим так, чтобы группа devops стала его основной:

```
Usermod -g devops victor
```

```
cat /etc/group
```

Обратите внимание, что произошло: мой пользователь теперь состоит в двух группах, а вот у нового пользователя группа единственная.

Удаление группы:

```
groupdel group
```

Ошибка! Пока в группе есть пользователи, её нельзя удалить. У пользователя всегда должна быть хотя бы одна группа.

Ну и в заключение научимся удалять пользователей и группы. Это очень просто. Для удаления пользователя есть команда:

```
userdel username
```

Обратите внимание, что домашняя директория пользователя не удалилась, и все его файлы сохранились. Чтобы её удалить, как вы помните, используется команда:

```
rm -r /home/username
```

Для того чтобы выполнить что-то от имени обычного пользователя, воспользуйтесь ключом -u.