

Права и группы. Linux

Сегодня разберемся с привелегиями пользователей. В любой ОС, да и не только, есть система прав и групп. В линукс при создании пользователя автоматически создается группа с именем этого пользователя и сам пользователь добавляется в свою группу. Можно посмотреть свои права и группы выполнив команду

```
$ id
```

вывод будет примерно таким

```
uid=1000(unit)                                gid=1000(unit)
группы=1000(unit),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),130(samb
ashare)
```

Тут мы видим id пользователя "**uid (User ID)**", id группы "**gid (Group ID)**", а также список групп к которым принадлежит данный пользователь. А вот вывод этой команды из под рута

```
uid=0(root) gid=0(root) группы=0(root)
```

Все по нулям, что говорит нам о полном доступе к системе. Переключиться на рута или выполнить команду от рута, можно:

```
$ sudo su
```

```
$ sudo <command>
```

Очень хорошо о запуске команд от другого пользователя написано в [этой статье](#), уж не стал копипастить и переводить.

Также список команд которыми ты можешь воспользоваться и какие софты юзать на машине определено группами, в которых состоит пользователь, и правами на файлы. В системе есть файл хранящий пароли пользователей, лежит он по пути "/etc/shadow". Для просмотра содержимого файла используется команда "cat"

```
$ cat /etc/shadow
```

Не получилось, да? Это потому, что на файл установлены права позволяющие читать и писать в него только владельцу и только читать на группу. Чтобы это увидеть можно воспользоваться командой "ls". Опять не получилось? Вывелся только путь? Не беда, как я уже говорил почти у каждой команды есть параметры, а узнать какие параметры принимает команда и что они делают можно 2-мя путями:

- Используя утилиту man. Выход из документации по нажатию "q".
\$ man ls
- Используя параметры "-h/--help". Есть противные скрипты которые перекрывают параметр "-h" используя его в своих целях, но оставляют справку по ключу "--help",
"ls" относится к таким.
\$ ls -h

Очень важно уметь находить и читать документацию. Теперь ознакомившись с маном ты можешь выполнить команду "ls" правильно, я понимаю, что тебе лень каждый раз

помнить и печатать пути, поэтому обратиться к истории команд тебе помогут, как это ни странно, клавиши вверх и вниз, а подставить ранее введенный параметр можно по шоткату <Alt+>>. Позже мы к этому вернемся.

А еще, если команда которую ты только что ввел, нужно запустить повторно из-под sudo то сделать можно так:

```
$ sudo !!
```

Чтобы не вводить в заблуждение "!!" команда повтора предыдущей команды!

Права на файл "/etc/shadow"

```
-rw-r----- 1 root shadow 1,4K апр 14 14:29 /etc/shadow
```

Первая колонка (*таблица выводимая "ls" разделяется пробелами*) - права на файл, вторая - кол-во ссылок на файл, владелец, группа, размер (в байтах), дата последнего изменения и имя файла. Отсюда мы видим, что файл принадлежит пользователю "root" и группе "shadow". А теперь перейдем к самим правам "-rw-r-----". Кол-во тире всегда одинаковое, сами тире означают, что права не установлены, а каждая буква дает определенное право. Дели права по 3 черточки (бита). Первый бит определяет тип объекта:

- | | | | |
|-----|---|---------------------|--------------------|
| • - | — | обычный | файл; |
| • d | — | | каталог; |
| • b | — | файл блочного | устройства; |
| • c | — | файл символического | устройства; |
| • s | — | доменное гнездо | (<i>socket</i>); |
| • p | — | именованный канал | (<i>pipe</i>); |
| • l | — | символическая | ссылка. |

В нашем случае - это файл. Далее идут права по 3 бита слева на право: владелец, группа, все остальные. Права обозначаются следующими буквами:

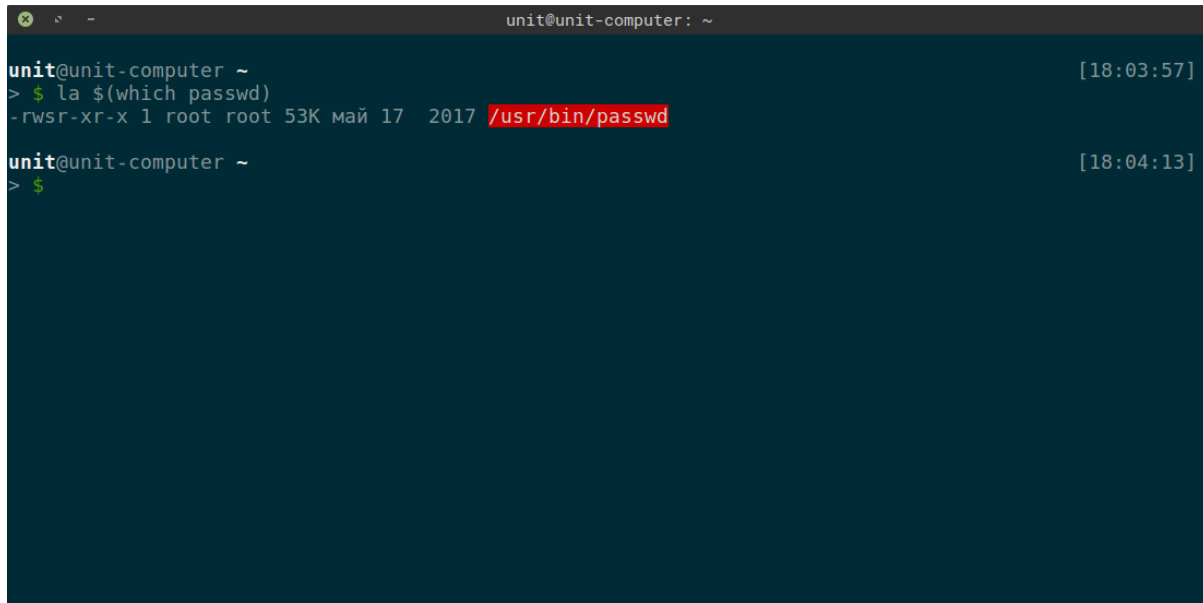
- r - право на чтение (**r**ead);
- w - право на запись (**w**rite);
- x - право на запуск (**e**xecute);

Таким образом первый набор бит ("rw-") дает владельцу файла ("root") право на чтение и запись данных в файл. Вторым - ("r--") право на чтение всем, кто входит в группу "shadow". И третий - ("---") никаких прав никому не дает. С правами в линуксе следующая история, что не разрешено - значит запрещено. Таким образом прочитать файл "/etc/shadow" можно так:

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

Ты, как любой пользователь системы, можешь поменять себе пароль, просто запустив команду "passwd", и дважды введя новый пароль. И тут должен возникнуть любопытный

вопрос: "Раз все пароли хранятся в `/etc/shadow`, а доступ на запись есть только у рута, то как я могу менять свой пароль с помощью `passwd`, ведь она запускается мной, а значит с моими правами?" Все верно, запускаемые скрипты и команды выполняются с привелегиями текущего пользователя и поэтому `cat` запущенный тобой не может прочитать файл `shadow`. А из-за необходимости давать обычным пользователям права которых у них не должно быть, были придуманы биты `suid` (*Set User ID или смена идентификатора пользователя*) и `sgid` (*Set Group ID или смена идентификатора группы*), которые на время запуска файла с таким битом предоставляют права владельца запускаемого файла. Посмотрим на права заданные для программы `passwd`



```
unit@unit-computer: ~  
[18:03:57]  
> $ la $(which passwd)  
-rwsr-xr-x 1 root root 53K май 17 2017 /usr/bin/passwd  
unit@unit-computer: ~  
[18:04:13]  
> $
```

Обрати внимание на первый набор бит `"rws"`, именно наличие бита `"s"` дает запускаемому процессу привелегии владельца файла и позволяет модифицировать файл с паролями, куда в обычной ситуации получить доступ нельзя.

Право `suid` считается потенциально опасным и неправильное его использование может вызвать проблемы. Например если бы `passwd` работал бы как `sudo` (без вопросов выполнял бы команды), то мы бы получили полный доступ к системе.

Для бита `suid` не предусмотрено отдельного места, поэтому это право перекрывает `"x"`, причем если есть право на запуск, то - `"s"`, если права на запуск нет, то - `"S"`.

Таким образом одним из векторов повышения привелегий может стать поиск исполняемых файлов, которые запускаются от `root` и куда есть доступ на запись (заглянуть можно, например, в файлы `cron`).

Также ты можешь менять права на объекты, владельцем которых являешься, с помощью команды `"chmod wXp /path/to"`, где:

`w` - указатель для кого устанавливаются права:

- `u` - для владельца
- `g` - для группы
- `o` - пользователи не входящие в группу
- `a` - для всех пользователей

`X` - указатель на модификатор:

- + - добавить право
- - - удалить право
- = - установить право

p - указатель на само право:

- r - чтение
- w - запись
- x - запуск

Вот несколько примеров использования команды `chmod`:

Предоставить всем пользователям право на выполнение:

```
$ chmod a+x /path/to
```

Удалить право на чтение и запись для всех, кроме владельца файла:

```
$ chmod go-rw /path/to
```

Дать всем права на чтение, запись и выполнение:

```
$ chmod ugo+rwX /path/to
```

Если опустить указание на то, кому предоставляется данное право, то подразумевается, что речь идет вообще обо всех пользователях, т. е. вместо

```
$ chmod a+x /path/to
```

можно записать просто

```
$ chmod +x /path/to
```

Рекурсивное применение прав для текущей директории. Удалить права на чтение, выполнение и запись во все файлы и каталоги текущей директории:

```
$ chmod -R go-rwx .
```

```
unit@unit-computer: ~/etc [21:08:58]
> $ ll
итого 2,0M
-rwxrwxr-x 1 unit unit 299 май 1 17:05 download_backups.sh
-rwx----- 1 unit unit 1,8M июл 15 2016 IDEALicenseServer
drwxrwxr-x 2 unit unit 4,0K май 1 17:13 shool_panel_backups
drwxrwxr-x 2 unit unit 4,0K май 1 17:10 user_panel_backups
drwxrwxr-x 2 unit unit 4,0K апр 18 11:37 vpn
drwx----- 7 unit unit 4,0K апр 2 14:18 weevely3-master
-rwx----- 1 unit unit 215K фев 25 13:02 weevely3.zip

unit@unit-computer ~/etc [21:09:00]
> $ chmod -R go-rwx .

unit@unit-computer ~/etc [21:09:18]
> $ ll
итого 2,0M
-rwx----- 1 unit unit 299 май 1 17:05 download_backups.sh
-rwx----- 1 unit unit 1,8M июл 15 2016 IDEALicenseServer
drwx----- 2 unit unit 4,0K май 1 17:13 shool_panel_backups
drwx----- 2 unit unit 4,0K май 1 17:10 user_panel_backups
drwx----- 2 unit unit 4,0K апр 18 11:37 vpn
drwx----- 7 unit unit 4,0K апр 2 14:18 weevely3-master
-rwx----- 1 unit unit 215K фев 25 13:02 weevely3.zip

unit@unit-computer ~/etc [21:09:19]
> $
```

```
unit@unit-computer: ~/etc [21:09:18]
> $ ll
итого 2,0M
-rwx----- 1 unit unit 299 май 1 17:05 download_backups.sh
-rwx----- 1 unit unit 1,8M июл 15 2016 IDEALicenseServer
drwx----- 2 unit unit 4,0K май 1 17:13 shool_panel_backups
drwx----- 2 unit unit 4,0K май 1 17:10 user_panel_backups
drwx----- 2 unit unit 4,0K апр 18 11:37 vpn
drwx----- 7 unit unit 4,0K апр 2 14:18 weevely3-master
-rwx----- 1 unit unit 215K фев 25 13:02 weevely3.zip

unit@unit-computer ~/etc [21:09:19]
> $ ll weevely3-master
итого 112K
drwx----- 4 unit unit 4,0K апр 2 14:18 bd
-rw----- 1 unit unit 2,5K ноя 24 01:36 CHANGELOG.md
drwx----- 3 unit unit 4,0K апр 2 14:39 core
-rw----- 1 unit unit 35K ноя 24 01:36 LICENSE
drwx----- 10 unit unit 4,0K апр 2 14:40 modules
-rw----- 1 unit unit 1,4K ноя 24 01:36 README.md
-rw----- 1 unit unit 57 ноя 24 01:36 requirements.txt
drwx----- 3 unit unit 4,0K апр 2 14:18 tests
drwx----- 3 unit unit 4,0K апр 2 14:40 utils
-rwx----- 1 unit unit 3,1K ноя 24 01:36 weevely.py

unit@unit-computer ~/etc [21:11:37]
> $
```

Тоже самое можно сделать так:

```
$ chmod -R go= .
```

При работе с правами можно лишиться привелегий, если удалить у себя право "x" на папку (все папки должны иметь это право для владельца).

Команду "find" разберем позже. Суть в том, что у директорий обязательно должно быть разрешение "x".

Можно задавать права через числовое представление этих прав. Т.е. каждое право имеет свое числовое представление:

- r - 4

- w - 2
- x - 1
- не давать никаких прав - 0

Чтобы предоставить права таким способом нужно сложить соответствующие цифры для каждого набора. Например: нам нужно чтобы запускать, писать и читать мог только владелец, читать и запускать участники группы, а все остальные только запускать. Тогда мы складываем $4+2+1=7$ (r+w+x), $4+1=5$ (r+x), $1=1$ (x). В итоге просто пишем результаты для каждой группы слитно - 751.

```
unit@unit-computer: /tmp/perms [10:39:02]
> $ ll
итого 0
-rw----- 1 unit unit 0 май 17 09:21 permfile

unit@unit-computer: /tmp/perms [10:39:04]
> $ chmod 751 permfile

unit@unit-computer: /tmp/perms [10:39:14]
> $ ll
итого 0
-rwxr-x--x 1 unit unit 0 май 17 09:21 permfile

unit@unit-computer: /tmp/perms [10:39:15]
> $
```

Этот способ используется чаще, хотя мне больше нравится первый.

Еще одна команда которую рассмотрим сегодня - это "chown", которая позволяет менять владельца и группу. Синтаксис тут следующий: "chown [OPTIONS] user:group /path/to", опции тут как у chmod с дополнениями, например вместо пользователя:группы можно указать файл от которого эти параметры унаследуются.

```
unit-computer# ls -l
итого 0
-rwxr-x--x 1 unit unit 0 май 17 09:21 permfile
unit-computer# chown :www-data permfile
unit-computer# ls -l
итого 0
-rwxr-x--x 1 unit www-data 0 май 17 09:21 permfile
unit-computer#
```

Тут я поменял группу у файла и теперь все кто входит в группу "www-data" могут читать и запускать этот файл. Важно помнить что из под обычного пользователя ты не сможешь поставить владельца или группу к которой не принадлежишь.

[illegible]