

Лабораторная работа №5

Цель работы

Изучение механизмов изменения идентификаторов, применения SetUID- и Sticky-битов. Получение практических навыков работы в консоли с дополнительными атрибутами. Рассмотрение работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

Задание

Часть 1

1. Войдите в систему от имени пользователя guest.
2. Создайте программу simpleid.c.
3. Скомпилируйте программу и убедитесь, что файл программы создан:
`gcc simpleid.c -o simpleid`
4. Выполните программу simpleid: `./simpleid`
5. Выполните системную программу `id`: `id` и сравните полученный вами результат с данными предыдущего пункта задания.
6. Усложните программу, добавив вывод действительных идентификаторов. Получившуюся программу назовите simpleid2.c.
7. Скомпилируйте и запустите simpleid2.c: `gcc simpleid2.c -o simpleid2` `./simpleid2`
8. От имени суперпользователя выполните команды:
`chown root:guest /home/guest/simpleid2`
`chmod u+s /home/guest/simpleid2`
9. Используйте `sudo` или повысьте временно свои права с помощью `su`. Поясните, что делают эти команды.
10. Выполните проверку правильности установки новых атрибутов и смены владельца файла simpleid2: `ls -l simpleid2`
11. Запустите simpleid2 и `id`: `./simpleid2` и `id`. Сравните результаты.
12. Проделайте тоже самое относительно SetGID-бита.
13. Создайте программу readfile.c:
14. Откомпилируйте её. `gcc readfile.c -o readfile`
15. Смените владельца у файла readfile.c (или любого другого текстового файла в системе) и измените права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог.
16. Проверьте, что пользователь guest не может прочитать файл readfile.c.
17. Смените у программы readfile владельца и установите SetU'D-бит.
18. Проверьте, может ли программа readfile прочитать файл readfile.c?
19. Проверьте, может ли программа readfile прочитать файл `/etc/shadow`?
Отразите полученный результат и ваши объяснения в отчёте.

Часть 2

1. Выясните, установлен ли атрибут Sticky на директории /tmp, для чего выполните команду `ls -l / | grep tmp`
2. От имени пользователя guest создайте файл file01.txt в директории /tmp со словом test: `echo "test" > /tmp/file01.txt`
3. Просмотрите атрибуты у только что созданного файла и разрешите чтение и запись для категории пользователей «все остальные»:
`ls -l /tmp/file01.txt`
`chmod o+rw /tmp/file01.txt`
`ls -l /tmp/file01.txt`
4. От пользователя guest2 (не являющегося владельцем) попробуйте прочитать файл /tmp/file01.txt: `cat /tmp/file01.txt`
5. От пользователя guest2 попробуйте дозаписать в файл /tmp/file01.txt слово test2 командой `echo "test2" > /tmp/file01.txt`. Удалось ли вам выполнить операцию?
6. Проверьте содержимое файла командой `cat /tmp/file01.txt`
7. От пользователя guest2 попробуйте записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой `echo "test3" > /tmp/file01.txt`. Удалось ли вам выполнить операцию?
8. Проверьте содержимое файла командой `cat /tmp/file01.txt`
9. От пользователя guest2 попробуйте удалить файл /tmp/file01.txt командой `rm /tmp/file01.txt`. Удалось ли вам удалить файл?
10. Повысьте свои права до суперпользователя следующей командой `su -` и выполните после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp: `chmod -t /tmp`
11. Покиньте режим суперпользователя командой `exit`
12. От пользователя guest2 проверьте, что атрибута t у директории /tmp нет: `ls -l / | grep tmp`
13. Повторите предыдущие шаги. Какие наблюдаются изменения?
14. Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем? Ваши наблюдения занесите в отчёт.
15. Повысьте свои права до суперпользователя и верните атрибут t на директорию /tmp: `su -, chmod +t /tmp, exit`

Теоретическое введение

В операционной системе Linux есть много отличных функций безопасности, но она из самых важных - это система прав доступа к файлам. Linux, как последователь идеологии ядра Linux в отличие от Windows, изначально проектировался как многопользовательская система, поэтому права доступа к файлам в linux продуманы очень хорошо. И это очень важно, потому что локальный доступ к файлам для всех программ и всех пользователей позволил бы вирусам без проблем уничтожить систему [1].

Расширенные атрибуты представляют собой пары имя: значение, которые постоянно связаны с файлами и каталогами, подобно тому как строки окружения связаны с процессом. Атрибут может быть определён или не определён. Если он определён, то его значение может быть или пустым, или не пустым [2].

Формат символьного режима: `+-=acdeijstuACDST` [3].

«+» обозначает добавление указанных атрибутов к существующим; [2]

«-» обозначает их снятие;

«=» обозначает установку только этих атрибутов файлам.

Символы «acdeijstuACDST» указывают на новые атрибуты файлов, некоторые атрибуты может назначить только суперпользователь (root): [3].

SUID - (сокращения от англ. set user ID upon execution — «установка ID пользователя во время выполнения») являются флагами прав доступа в Unix, которые разрешают пользователям запускать исполняемые файлы с правами владельца исполняемого файла. Иногда файлы требуют разрешения на выполнение для пользователей, которые не являются членами группы владельца, в этом случае вам потребуется предоставить специальные разрешения на выполнение. Когда SUID установлен, пользователь может запускать любую программу, такую как владелец программы.[4].

```
chmod u+s {filename}
```

GUID - (сокращения от англ. set group ID upon execution — «установка ID группы во время выполнения») являются флагами прав доступа в Unix, которые разрешают пользователям запускать исполняемые файлы с правами группы исполняемого файла.[4].

```
chmod g+s {filename}
```

Так же, как SUID, установив SGID бит для файла он устанавливает ваш идентификатор группы для группы файла в то время как файл выполняется. Это действительно полезно в случае когда у вас есть реальные установки в многопользовательском режиме где у пользователей есть доступ к файлом. В одной домашней категории я действительно не нашел использования для SGID. Но основная концепция является такой же, как и у SUID, файлы у которых SGID бит устанавливается, то они принадлежат к этой группе, а не к этому пользователю.[4].

Sticky Bit используется в основном для каталогов, чтобы защитить в них файлы. Из такого каталога пользователь может удалить только те файлы, владельцем которых он является. Примером может служить каталог /tmp, в который запись открыта для всех пользователей, но нежелательно удаление чужих файлов.[5].

```
chmod +t {filename}
```

Оборудование

Лабораторная работа выполнялась дома со следующими характеристиками техники:

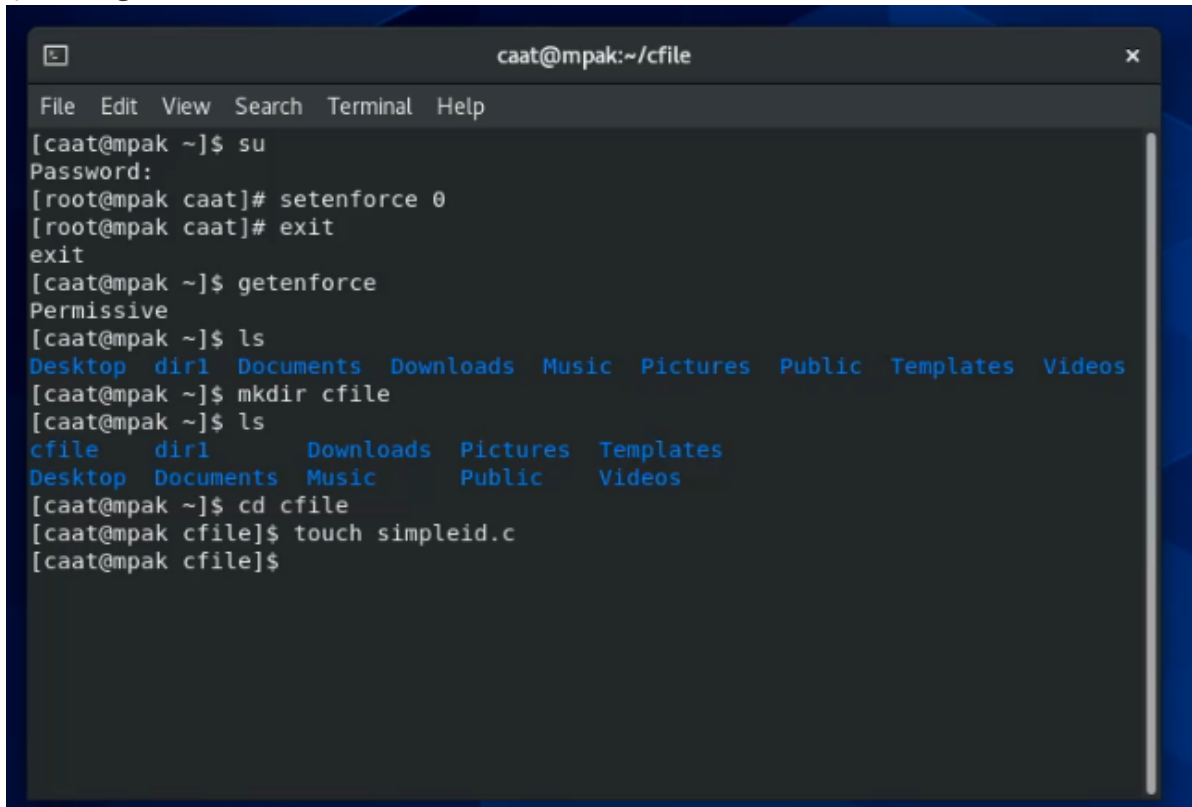
- Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.81GHz
- ОС Майкрософт Windows 10
- VirtualBox верс. 6.1.26

Выполнение лабораторной работы

Часть 1

1. Вошла в систему от имени пользователя saat. Даю себе прова root и отключаю систему запретов. Проверяю данный факт, и выхожу из режима суперпользователя.

(рис. -@fig:001)



```
caat@mpak:~/cfile
File Edit View Search Terminal Help
[caat@mpak ~]$ su
Password:
[root@mpak caat]# setenforce 0
[root@mpak caat]# exit
exit
[caat@mpak ~]$ getenforce
Permissive
[caat@mpak ~]$ ls
Desktop dirl Documents Downloads Music Pictures Public Templates Videos
[caat@mpak ~]$ mkdir cfile
[caat@mpak ~]$ ls
cfile dirl Downloads Pictures Templates
Desktop Documents Music Public Videos
[caat@mpak ~]$ cd cfile
[caat@mpak cfile]$ touch simpleid.c
[caat@mpak cfile]$
```

2. Создаю файл simpleid.c и пишу в нем программу по образцу из лабораторной работы 5.

(рис. -@fig:002)



```
*simpleid.c
~/cfile
Save

#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int
main()
{
    uid_t uid = geteuid();
    gid_t gid = getegid();
    printf ("uid=%d, gid=%d\n", uid, gid);
    return 0;
}
```

3. Скомпилировала программу и убедилась, что файл программы создан. Как видно, теперь у нас есть исполняемый файл simpleid, который подсвечивается зеленым.

(рис. -@fig:003)

```
[caat@mpak ~]$ cd cfile
[caat@mpak cfile]$ touch simpleid.c
[caat@mpak cfile]$ gcc simpleid.c -o simpleid
gcc: error: simpleid.c: No such file or directory
gcc: fatal error: no input files
compilation terminated.
[caat@mpak cfile]$ gcc simpleid.c -o simpleid
[caat@mpak cfile]$ ls
simpleid  simpleid.c
[caat@mpak cfile]$ ./s
```

4-5. Выполнила программу ./simpleid и id. Результаты они вывели одинаковые, так как обе программы просто выводят id пользователя, которому принадлежит файл и id группы файла.

(рис. -@fig:004)

```
[caat@mpak cfile]$ gcc simpleid.c -o simpleid
[caat@mpak cfile]$ ls
simpleid  simpleid.c
[caat@mpak cfile]$ ./simpleid
uid=1002, gid=1003
[caat@mpak cfile]$ id
uid=1002(caat) gid=1003(caat) groups=1003(caat) context=unconfined_u:unconfined_
r:unconfined_t:s0-s0:c0.c1023
[caat@mpak cfile]$
```

6. Создала новый файл simpleid2.c, добавив вывод действительных идентификаторов.

(рис. -@fig:005)

```
Open  simpleid2.c
~/cfile

#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int
main()
{
    uid_t real_uid = getuid();
    uid_t e_uid = geteuid();

    gid_t real_gid = getgid();
    gid_t e_gid = getegid();

    printf ("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
    printf ("real_uid=%d, real_gid=%d\n", real_uid, real_gid);

    return 0;
}
```

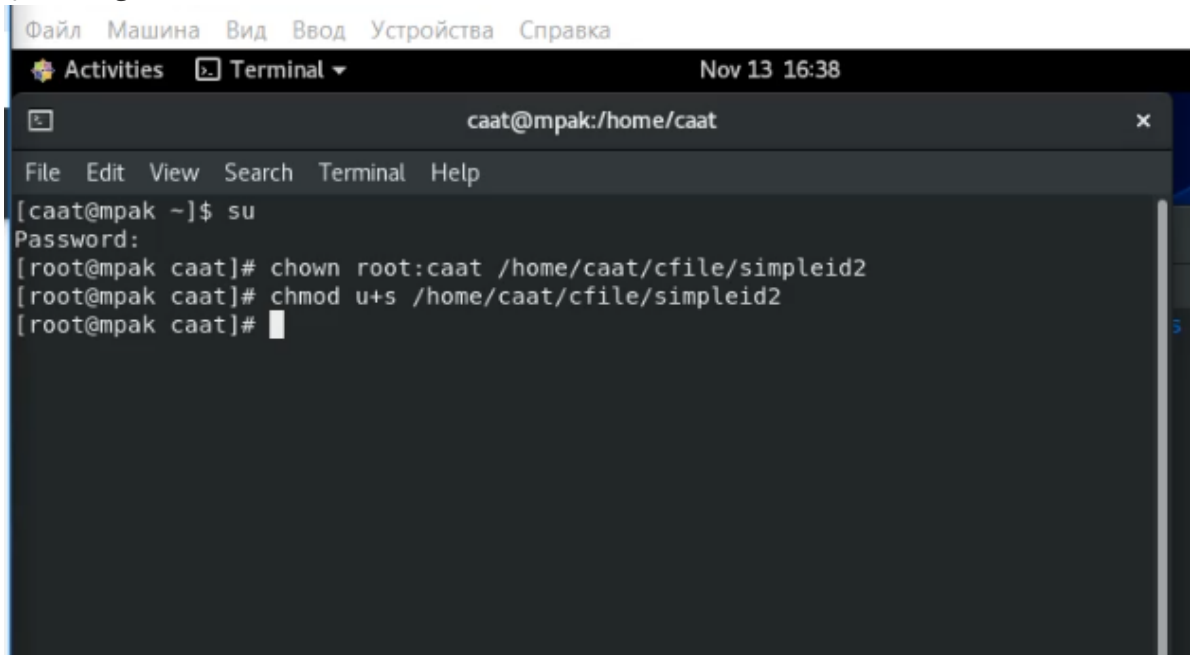
7. Скомпилировала и запустила simpleid2.c.

(рис. -@fig:006)

```
[caat@mpak cfile]$ ./simpleid
uid=1002, gid=1003
[caat@mpak cfile]$ id
uid=1002(caat) gid=1003(caat) groups=1003(caat) context=unconfined_u:unconfined_
r:unconfined_t:s0-s0:c0.c1023
[caat@mpak cfile]$ gcc simpleid2.c -o simpleid2
[caat@mpak cfile]$ ./simpleid2
e_uid=1002, e_gid=1003
real_uid=1002, real_gid=1003
[caat@mpak cfile]$
```

8-9. От имени суперпользователя выполнила команды: `chown root:caat simpleid2` и `chmod u+s simpleid2`

(рис. -@fig:007)



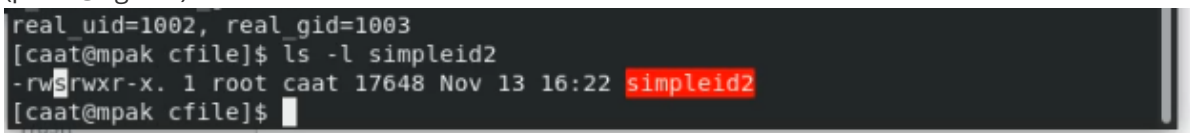
```
Файл Машина Вид Ввод Устройства Справка
Activities Terminal Nov 13 16:38
caat@mpak:/home/caat
File Edit View Search Terminal Help
[caat@mpak ~]$ su
Password:
[root@mpak caat]# chown root:caat /home/caat/cfile/simpleid2
[root@mpak caat]# chmod u+s /home/caat/cfile/simpleid2
[root@mpak caat]#
```

Первая команда меняет владельца файла `simpleid2.c` на `root`, а группа у файла теперь `caat`. Т.е. теперь владельцем файла является суперпользователь, а запускать его смогут и суперпользователь и группа `caat`.

Вторая команда устанавливает дополнительное право `UID`. Теперь все кто запускают файл на время работы файла получают права владельца этого самого файла. Т.е. если программа считывает права пользователя для работы, то программа будет видеть, что у пользователя который ее запустил теперь права `root`.

10. Выполнила проверку правильности установки новых атрибутов и смены владельца файла `simpleid2`: `ls -l simpleid2`

(рис. -@fig:008)

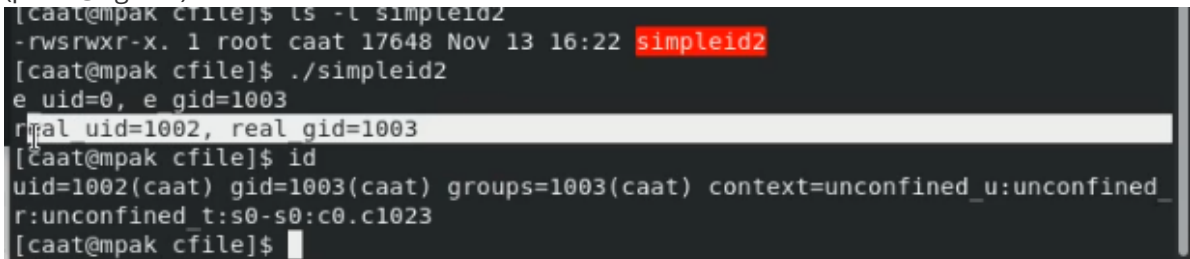


```
real_uid=1002, real_gid=1003
[caat@mpak cfile]$ ls -l simpleid2
-rwsrwxr-x. 1 root caat 17648 Nov 13 16:22 simpleid2
[caat@mpak cfile]$
```

Как видно на картинке, в атрибутах у владельца на месте `X` теперь появилась буква `S`. Также команда показывает что пользователь `root` теперь владелец файла, а группа у файла `caat`.

11. Запустила `simpleid2` и `id: ./simpleid2` и `id`.

(рис. -@fig:009)



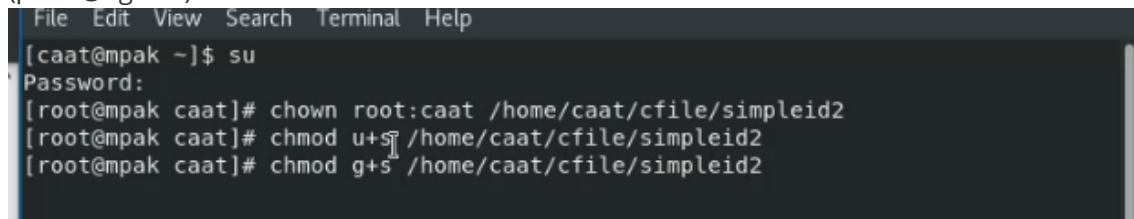
```
[caat@mpak cfile]$ ls -l simpleid2
-rwsrwxr-x. 1 root caat 17648 Nov 13 16:22 simpleid2
[caat@mpak cfile]$ ./simpleid2
e_uid=0, e_gid=1003
real_uid=1002, real_gid=1003
[caat@mpak cfile]$ id
uid=1002(caat) gid=1003(caat) groups=1003(caat) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[caat@mpak cfile]$
```

`e_uid` показывает какие права пользователя использует программа. Программа пользуется правами суперюзера (думает, что программу запустил `root`). Это значение поменялось относительно псолендного запуска, т.к. мы добавили права `u+s`. Тоже самое и с `e_gid` `real_uid` указывает фактического пользователя, который запустил процесс. Запустил его `caat`. Тоже самое и с `real_gid`

id показывает те же самые результаты, только нигде не показывает информацию о правах суперюзера.

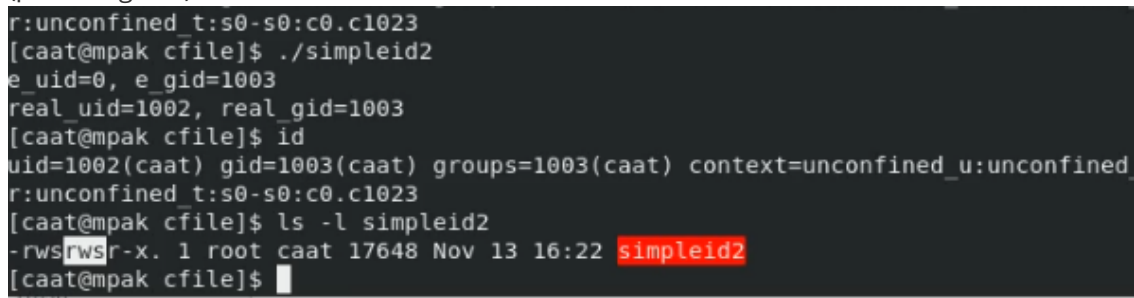
12. Прodelала тоже самое относительно SetGID-бита. Т.е установила на файл права g+s. Запустила файл и команду id. Команды снова вывели одинаковую непротиворечащую информацию.

(рис. -@fig:010)



```
File Edit View Search Terminal Help
[caat@mpak ~]$ su
Password:
[root@mpak caat]# chown root:caat /home/caat/cfile/simpleid2
[root@mpak caat]# chmod u+s /home/caat/cfile/simpleid2
[root@mpak caat]# chmod g+s /home/caat/cfile/simpleid2
```

(рис. -@fig:011)

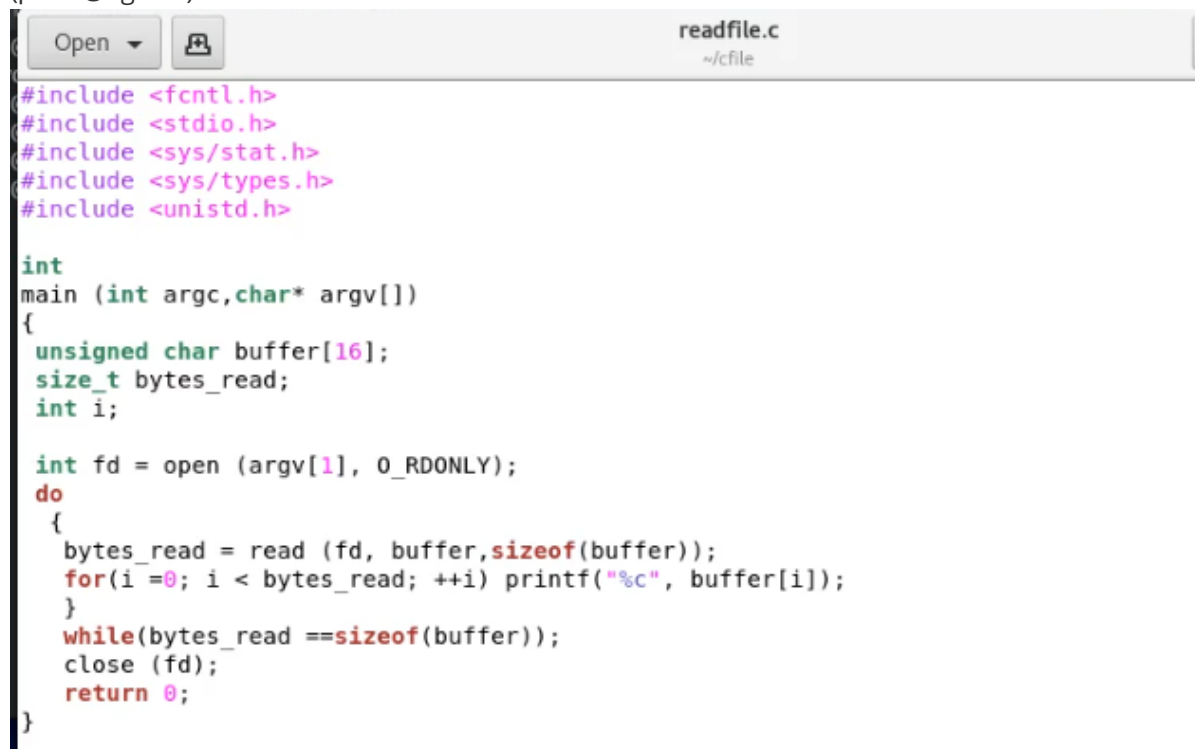


```
r:unconfined t:s0-s0:c0.c1023
[caat@mpak cfile]$ ./simpleid2
e_uid=0, e_gid=1003
real_uid=1002, real_gid=1003
[caat@mpak cfile]$ id
uid=1002(caat) gid=1003(caat) groups=1003(caat) context=unconfined_u:unconfined
r:unconfined t:s0-s0:c0.c1023
[caat@mpak cfile]$ ls -l simpleid2
-rwsrwsr-x. 1 root caat 17648 Nov 13 16:22 simpleid2
[caat@mpak cfile]$
```

Как можно увидеть на последней картинке, у файла действительно установились права g+s, поэтому теперь на месте X можно увидеть S

- 13-14. Создала программу readfile.c. Откомпилировала её. gcc readfile.c -o readfile

(рис. -@fig:012)

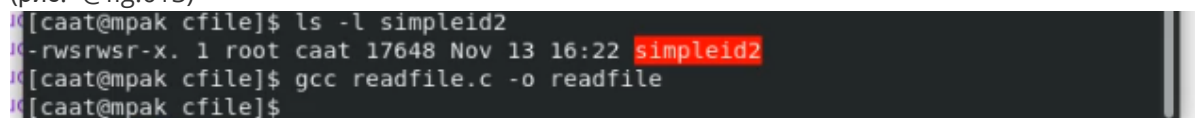


```
Open [readfile.c] ~/cfile
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int
main (int argc, char* argv[])
{
    unsigned char buffer[16];
    size_t bytes_read;
    int i;

    int fd = open (argv[1], O_RDONLY);
    do
    {
        bytes_read = read (fd, buffer, sizeof(buffer));
        for(i = 0; i < bytes_read; ++i) printf("%c", buffer[i]);
    }
    while(bytes_read == sizeof(buffer));
    close (fd);
    return 0;
}
```

(рис. -@fig:013)



```
[caat@mpak cfile]$ ls -l simpleid2
-rwsrwsr-x. 1 root caat 17648 Nov 13 16:22 simpleid2
[caat@mpak cfile]$ gcc readfile.c -o readfile
[caat@mpak cfile]$
```

- 15-16. Сменила владельца у файла readfile.c командой chown root:root и изменила права у файла.

(рис. -@fig:014)

```
File Edit View Search Terminal Help
[caat@mpak ~]$ su
Password:
[root@mpak caat]# chown root:caat /home/caat/cfile/simpleid2
[root@mpak caat]# chmod u+s /home/caat/cfile/simpleid2
[root@mpak caat]# chmod g+s /home/caat/cfile/simpleid2
[root@mpak caat]# chown root:root /home/caat/cfile/readfile.c
[root@mpak caat]# chmod go-r /home/caat/cfile/readfile.c
```

(рис. -@fig:015)

```
[caat@mpak cfile]$ ls -l readfile.c
-rw-r--r--. 1 root root 422 Nov 13 16:57 readfile.c
[caat@mpak cfile]$ ls -l readfile.c
-rw-----. 1 root root 422 Nov 13 16:57 readfile.c
[caat@mpak cfile]$ cat readfile.c
cat: readfile.c: Permission denied
[caat@mpak cfile]$
```

Теперь только пользователь root может читать этот файл, а пользователь caat получает отказ в операции чтение.

17-19. Сменила у программы readfile владельца и установите SetU'D-бит.

(рис. -@fig:016)

```
[root@mpak caat]# chown root:root /home/caat/cfile/readfile.c
[root@mpak caat]# chmod go-r /home/caat/cfile/readfile.c
[root@mpak caat]# chown root:root /home/caat/cfile/readfile
[root@mpak caat]# chmod u+s /home/caat/cfile/readfile
[root@mpak caat]#
```

На нижнем слайде мы видим как программа читает файл readfile.c, а затем и /etc/shadow.

(рис. -@fig:017)

```
[caat@mpak cfile]$ ./readfile readfile.c
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int
main (int argc, char* argv[])
{
    unsigned char buffer[16];
    size_t bytes_read;
    int i;

    int fd = open (argv[1], O_RDONLY);
    do
    {
        bytes_read = read (fd, buffer, sizeof(buffer));
        for(i=0; i < bytes_read; ++i) printf("%c", buffer[i]);
    }
    while(bytes_read == sizeof(buffer));
    close (fd);
    return 0;
}
[caat@mpak cfile]$ ./readfile /etc/shadow
root:$6$NKVXPM5T1jrX.FXZ$KhYx2KDxA6qu0z/Uzp/sVf2n92DwG5HeUI3A6yvTm/00792bCUCWj0/Tly9MJ5ZYV8L48J.Yv1WJS.E
qN7ugE0::0:99999:7:::
bin:*:18397:0:99999:7:::
daemon:*:18397:0:99999:7:::
adm:*:18397:0:99999:7:::
lp:*:18397:0:99999:7:::
sync:*:18397:0:99999:7:::
shutdown:*:18397:0:99999:7:::
halt:*:18397:0:99999:7:::
mail:*:18397:0:99999:7:::
operator:*:18397:0:99999:7:::
games:*:18397:0:99999:7:::
ftp:*:18397:0:99999:7:::
nobody:*:18397:0:99999:7:::
dbus:!:18899::::::
systemd-coredump:!:18899::::::
systemd-resolve:!:18899::::::
tcpd:!:18899::::::
```


Так как мы дали нашему исполняемому файлу особый атрибут S, теперь этот файл будет пользоваться правами суперюзера. На нескольких пунктах до этого мы изменили владельца файла readfile.c, т.е. теперь его может читать только root. Так как наш скрипт readfile пользовался правами суперюзера, он и мог прочитать нужный нам файл readfile.c. То же самое и произошло с /etc/shadow, так как проводить операции с ним может только суперюзер.

Часть 2

1. Выясняем, установлен ли атрибут Sticky на директории /tmp, для чего выполнила команду `ls -l / | grep tmp`

(рис. -@fig:018)

```
[caat@mpak cfile]$ ls -l | grep tmp
[caat@mpak cfile]$ ls -l / | grep tmp
drwxrwxrwt. 16 root root 4096 Nov 13 17:13 tmp
[caat@mpak cfile]$
```

Атрибут t действительно установлен на папке tmp. Она защищена от передвижения и удаления из директории, так как она является системной директорией.

- 2-3. От имени пользователя caat создала файл file01.txt в директории /tmp со словом test.

(рис. -@fig:019)

```
[caat@mpak cfile]$ cd /tmp
[caat@mpak tmp]$ echo "tets"> /tmp/file01.txt
[caat@mpak tmp]$ ls -l /tmp/file01.txt
-rw-rw-r--. 1 caat caat 5 Nov 13 17:23 /tmp/file01.txt
[caat@mpak tmp]$ chmod o+rw file01.txt
[caat@mpak tmp]$ ls -l /tmp/file01.txt
-rw-rw-rw-. 1 caat caat 5 Nov 13 17:23 /tmp/file01.txt
[caat@mpak tmp]$
```

Далее посмотрела атрибуты у только что созданного файла (не было разрешения на запись для "всех остальных") и разрешила чтение и запись для категории пользователей «все остальные». Атрибуты установились правильно. Теперь у всех групп пользователей есть права на чтение и запись в файл.

- 4-9. От пользователя caat (не являющегося владельцем и не состоящем в общей группе) попробую прочитать файл `cat /tmp/file01.txt`. Файл спокойно читается.

(рис. -@fig:020)

```
[root@mpak caat]# su guest
[guest@mpak caat]$ cat /tmp/file01.txt
tets
[guest@mpak caat]$ cat /tmp/file01.txt echo "tets2">/tmp/file01.txt
cat: echo: Permission denied
cat: tets2: Permission denied
[guest@mpak caat]$ echo "tets2">/tmp/file01.txt
[guest@mpak caat]$ cat /tmp/file01.txt
tets2
[guest@mpak caat]$ echo "tets3">>/tmp/file01.txt
[guest@mpak caat]$ cat /tmp/file01.txt
tets2
tets3
[guest@mpak caat]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest@mpak caat]$
```

Попробовала перезаписать слово test3, стерев при этом всю имеющуюся в файле информацию командой `echo "test2" > /tmp/file01.txt`. Эта операция удалась.

Попробую дозаписать в файл /tmp/file01.txt слово test2 командой `echo "test2" >> /tmp/file01.txt`. Дозаписать удалось.

Попробовала удалить файл /tmp/file01.txt командой `rm /tmp/file01.txt`. Файл удалить не удалось.

10-12. Повысила свои права до суперпользователя и сняла атрибут `t` (Sticky-бит) с директории /tmp. Покинула режим суперпользователя командой `exit`.

(рис. -@fig:021)

```
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest@mpak caat]$ su
Password:
[root@mpak caat]# cmod -t /tmp
bash: cmod: command not found...
Similar command is: 'kmod'
[root@mpak caat]# chmod -t /tmp
[root@mpak caat]# exit
exit
[guest@mpak caat]$ ls -l / | grep tmp
drwxrwxrwx. 16 root root 4096 Nov 13 17:30 tmp
[guest@mpak caat]$
```

От пользователя `caat` проверила, что атрибут снялся у директории. Его действительно теперь нет, а на его месте `X` у группы "все остальные".

13-14. Повторяю предыдущие действия без атрибута `t`.

(рис. -@fig:022)

```
[guest@mpak caat]$ su
Password:
[root@mpak caat]# chmod +t /tmp
[root@mpak caat]# exit
exit
[guest@mpak caat]$ ls -l / | grep tmp
drwxrwxrwt. 16 root root 4096 Nov 13 17:35 tmp
[guest@mpak caat]$
```

Чтение файла работает.

Дозапись в файл работает.

Дозапись в файл со стиранием предыдущей информации - работает

Файл удалить удалось.

Это связано с тем, что атрибут `t` до этого предотвращал передвигание и удаление файлов из директории `tmp`. Как только мы сняли атрибут, удаление файла стало возможным.

15. Повыла свои права до суперпользователя и вернула атрибут `t` на директорию /tmp.

(рис. -@fig:023)

```
[guest@mpak caat]$ su
Password:
[root@mpak caat]# chmod +t /tmp
[root@mpak caat]# exit
exit
[guest@mpak caat]$ ls -l / | grep tmp
drwxrwxrwt. 16 root root 4096 Nov 13 17:35 tmp
[guest@mpak caat]$
```

Не забыла выйти из режима суперпользователя.

Выводы

Изучила механизм изменения идентификаторов, применения SetUID- и Sticky-битов. Получила практические навыки работы в консоли с дополнительными атрибутами. Рассмотрела работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

Список литературы

1. CentOS // Википедия URL: <https://ru.wikipedia.org/wiki/CentOS> (дата обращения: 10.11.2021).
2. Права Доступа // LoST // <https://losst.ru/prava-dostupa-k-fajlam-v-linux> (дата обращения: 11.11.2021).
3. Расширенные Атрибуты // linux-notes URL: <https://linux-notes.org/izmenenie-atributov-flagov-na-fajlah-v-unix-linux/> (дата обращения: 9.11.2021).
4. Стандартные права (SUID, SGID, Sticky bit) в Unix/Linux // LINUX-NOTES.ORG URL: <https://linux-notes.org/standartny-e-prava-unix-suid-sgid-sticky-bit/> (дата обращения: 12.11.2021).
5. Sticky bit // Wikipedia URL: https://ru.wikipedia.org/wiki/Sticky_bit (дата обращения: 12.11.2021).