

Лабораторная работа № 5. Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Дисциплина: Информационная безопасность

Манаева Варвара Евгеньевна.

07 октября 2023

Российский университет дружбы народов, Москва, Россия

Цели и задачи 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.

- Настроить по заданию лабораторный стенд;
- Создать программы;
- Подробно разобрать влияние Sticky-бита на удаление и запись файлов.

Выполнение лабораторной работы

Создание программ

Войдём в систему от имени пользователя guest.

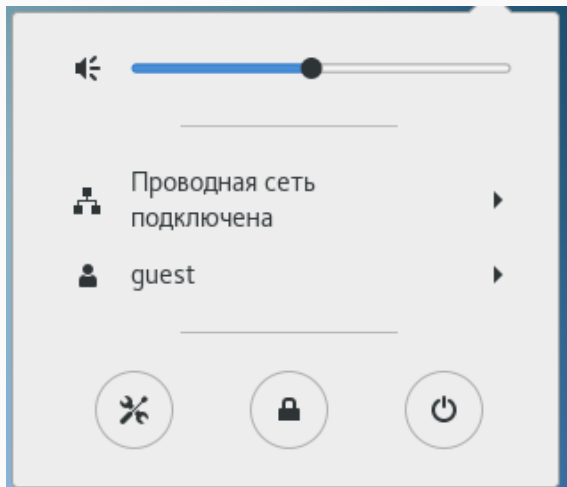


Рис. 1: Выполнен вход

Создадим программу simpleid.c:

```
GNU nano 2.3.1                               Файл: simpleid.c
#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;
}
```

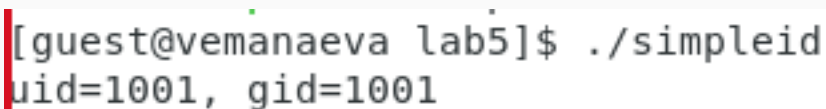
Рис. 2: Создана программа

Скомпилируем программу и убедимся, что файл программы создан, командой `gcc simpleid.c -o simpleid`

```
[guest@vemanaeva lab5]$ gcc simpleid.c -o simpleid
[guest@vemanaeva lab5]$ ls -a
.  ..  simpleid  simpleid.c
[guest@vemanaeva lab5]$
```

Рис. 3: Компилируем программу

Выполним программу simpleid командой `./simpleid`

A terminal window with a light gray background. The prompt is [guest@vemanaeva lab5]\$ and the command ./simpleid has been entered. The output is uid=1001, gid=1001. A red vertical bar is on the left side of the terminal window.

```
[guest@vemanaeva lab5]$ ./simpleid
uid=1001, gid=1001
```

Рис. 4: Вывод программы simpleid

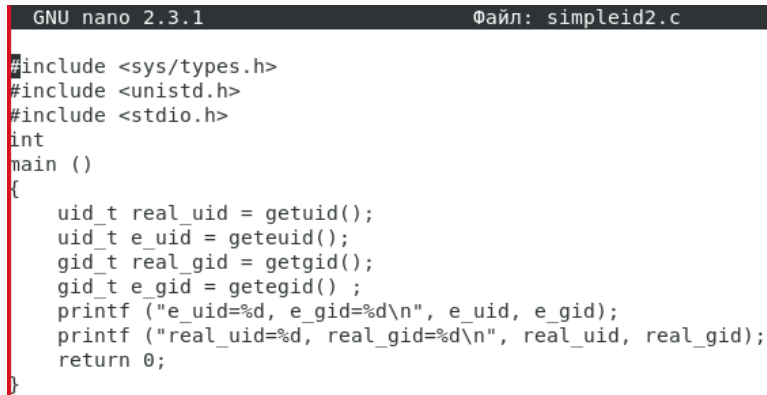
Выполним системную программу `id`

```
[guest@vema-naeva lab5]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 5: Вывод системной программы

При использовании команды `id`, в отличие от использования написанной нами программы, выводится не только номер пользователя и номер группы, но и имя пользователя и название группы (в скобках после соответствующих номеров), а также контекст.

Усложним программу, добавив вывод действительных идентификаторов, и получившуюся программу назовём `simpleid2.c`.



```
GNU nano 2.3.1                                Файл: simpleid2.c
#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;
}
```

Рис. 6: Создание второй программы

Скомпилируем и запустим `simpleid2.c` командами:

```
[guest@vemanaeva lab5]$ gcc simpleid2.c -o simpleid2  
[guest@vemanaeva lab5]$ ./simpleid2  
e_uid=1001, e_gid=1001  
real_uid=1001, real_gid=1001  
[guest@vemanaeva lab5]$
```

Рис. 7: Компиляция и запуск второй программы

Попробуем выполнить команды:

```
[guest@vemanaeva lab5]$ chown root:guest simpleid2  
chown: изменение владельца «simpleid2»: Операция не позволена
```

Рис. 8: Не получается выполнить команды

Повысим временно свои права с помощью su и выполним эти команды.

```
[guest@vemanaeva lab5]$ su -  
Пароль:  
Последний вход в систему:Сб сен 30 17:50:36 MSK 2023на pts/1  
[root@vemanaeva ~]# chown root:guest /home/guest/simpleid2  
chown: невозможно получить доступ к «/home/guest/simpleid2»: Нет такого файла или каталога  
[root@vemanaeva ~]# chown root:guest /home/guest/lab5/simpleid2  
[root@vemanaeva ~]# chmod u+s /home/guest/lab5/simpleid2  
[root@vemanaeva ~]# █
```

Рис. 9: Меняем пользователя на суперпользователя и выполняем команды

Команда `chown root:guest /home/guest/lab5/simpleid2` меняет владельца каталога на **root**, а группу каталога меняет на группу **guest** (вернее, оставляет без изменений).
Команда `chmod u+s /home/guest/lab5/simpleid2` даёт права на выполнение от имени суперпользователя.

Выполним проверку правильности установки новых атрибутов и смены владельца файла `simpleid2` командой `ls -l simpleid2`

```
[guest@vemanaeva lab5]$ ls -l simpleid2  
-rwsrwxr-x. 1 root guest 8616 окт  7 13:40 simpleid2
```

Рис. 10: Проверка

Запустим simpleid2 и id

```
[guest@vemanaeva lab5]$ ./simpleid2  
e_uid=0, e_gid=1001  
real_uid=1001, real_gid=1001  
[guest@vemanaeva lab5]$ id  
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:  
c0.c1023
```

Рис. 11: Выполняем программу и системную программу

Команда `id`, в отличие от написанной программы, выводит только информацию о пользователе, который запросил запуск системной программы, в то время как `simpleid2` выводит информацию и о владельце файла, и о том, кто его запускает.

Прделаем тоже самое относительно SetGID-бита.

```
[root@vemanaeva ~]# chown root:root /home/guest/lab5/simpleid2
[root@vemanaeva ~]# chmod g+s /home/guest/lab5/simpleid2
[root@vemanaeva ~]# su guest
[guest@vemanaeva root]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@vemanaeva root]$ cd ~/lab5
[guest@vemanaeva lab5]$ ./simpleid2
e_uid=1001, e_gid=0
real_uid=1001, real_gid=1001
```

Рис. 12: Повторение действий с SetGID

Создадим программу readfile.c

```
GNU nano 2.3.1                                Файл: 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;
}
```

Рис. 13: Создаём программу

Откомпилируем её командой `gcc readfile.c -o readfile`

```
[guest@vemanaeva lab5]$ gcc readfile.c -o readfile
[guest@vemanaeva lab5]$ ls -a
.  ..  readfile  readfile.c  simpleid  simpleid2  simpleid2.c  simpleid.c
[guest@vemanaeva lab5]$
```

Рис. 14: Компилируем программу и проверяем наличие

Сменим владельца у файла readfile.c (или любого другого текстового файла в системе) и изменим права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог.

```
[root@vemanaeva ~]# chown root:root /home/guest/lab5/readfile.c
[root@vemanaeva ~]# chmod 770 /home/guest/lab5/readfile.c
[root@vemanaeva ~]# ls -l /home/guest/lab5/readfile.c
-rwxrwx---. 1 root root 458 окт  7 14:00 /home/guest/lab5/readfile.c
[root@vemanaeva ~]#
```

Рис. 15: Изменяем права на файл

Проверим, что пользователь guest не может прочитать файл `readfile.c` командой `cat /home/guest/lab5/readfile.c`

```
[guest@vemanaeva lab5]$ cat readfile.c  
cat: readfile.c: Отказано в доступе
```

Рис. 16: Проверка

Сменим у программы readfile владельца и установим SetUID-бит.

```
[root@vemanaeva ~]# chown root:root /home/guest/lab5/readfile
[root@vemanaeva ~]# chmod u+s /home/guest/lab5/readfile
[root@vemanaeva ~]# ls -l /home/guest/lab5/readfile
-rwsrwxr-x. 1 root root 8552 окт  7 14:01 /home/guest/lab5/readfile
[root@vemanaeva ~]#
```

Рис. 17: Меняем владельца

Проверим, может ли программа readfile прочитать файл readfile.c

```
[guest@vemanaeva lab5]$ ./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;
}
[guest@vemanaeva lab5]$
```

Рис. 18: Проверяем возможность чтения файла `readfile.c`

Проверим, может ли программа readfile прочитать файл /etc/shadow

```
[guest@vemanaeva lab5]$ ./readfile /etc/shadow
root:$6$sNDsC5pnY3mm94V7$0ek8eynNXXY1vHJAi3rNFm8VE9gRCtdwlcRS.XEUa0DacZJg1LvaUpaCqrI.03e5NLD9yyb..Fc00QTM3Wvs0::0:99999:7:::
bin:!:18353:0:99999:7:::
daemon:!:18353:0:99999:7:::
adm:!:18353:0:99999:7:::
lp:!:18353:0:99999:7:::
sync:!:18353:0:99999:7:::
shutdown:!:18353:0:99999:7:::
halt:!:18353:0:99999:7:::
mail:!:18353:0:99999:7:::
operator:!:18353:0:99999:7:::
games:!:18353:0:99999:7:::
ftp:!:18353:0:99999:7:::
nobody:!:18353:0:99999:7:::
systemd-network:!!:19609::::::
dbus:!!:19609::::::
polkitd:!!:19609::::::
libstoragemgmt:!!:19609::::::
colord:!!:19609::::::
rpc:!!:19609:0:99999:7:::
saned:!!:19609::::::
```

Рис. 19: Проверяем возможность чтения файла /etc/shadow

Да, программа может читать данный файл.

Исследование Sticky-бита

Выясним, установлен ли атрибут `Sticky` на директории `/tmp` командой `ls -l / | grep tmp`

```
[guest@vemanaeva lab5]$ ls -l / | grep tmp  
drwxrwxrwt. 23 root root 4096 окт  7 14:08 tmp
```

Рис. 20: Sticky-бит установлен на директории `/tmp`

От имени пользователя guest создадим файл file01.txt в директории /tmp со словом test командой `echo "test" > /tmp/file01.txt`

```
[guest@vemanaeva lab5]$ echo "test" > /tmp/file01.txt  
[guest@vemanaeva lab5]$ cat /tmp/file01.txt  
test  
[guest@vemanaeva lab5]$
```

Рис. 21: Создадим файл

Просмотрим атрибуты у только что созданного файла и разрешим чтение и запись для категории пользователей «все остальные» командами:

```
[guest@vemanaeva lab5]$ ls -l /tmp/file01.txt
-rw-rw-r--. 1 guest guest 5 окт  7 14:14 /tmp/file01.txt
[guest@vemanaeva lab5]$ chmod o+rw /tmp/file01.txt
[guest@vemanaeva lab5]$ rw /tmp/file01.txt
bash: rw: команда не найдена...
[guest@vemanaeva lab5]$ [guest@vemanaeva lab5]$
bash: [guest@vemanaeva: команда не найдена...
[guest@vemanaeva lab5]$ ls -l /tmp/file01.txt
-rw-rw-rw-. 1 guest guest 5 окт  7 14:14 /tmp/file01.txt
[guest@vemanaeva lab5]$
```

Рис. 22: Выдаём права на запись категории пользователей “все остальные”

От пользователя `guest2` (не являющегося владельцем) попробуем прочитать файл `/tmp/file01.txt` командой `cat /tmp/file01.txt`

```
[guest2@vemanaeva ~]$ cat /tmp/file01.txt  
test
```

Рис. 23: Смотрим содержимое файла

От пользователя `guest2` попробуем дозаписать в файл `/tmp/file01.txt` слово `test2` командой `echo "test2" >> /tmp/file01.txt`

```
[guest2@vemanaeva ~]$ echo "test2" >> /tmp/file01.txt
```

Рис. 24: Дописываем информацию в файл

Операцию выполнить удалось.

Проверим содержимое файла командой `cat /tmp/file01.txt`

```
[guest2@vemanaeva ~]$ cat /tmp/file01.txt  
test  
test2  
[guest2@vemanaeva ~]$
```

Рис. 25: Смотрим содержимое файла

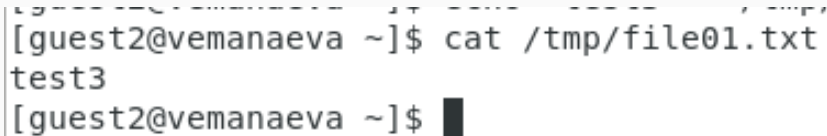
От пользователя guest2 попробуем записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой echo "test3" > /tmp/file01.txt

```
[guest2@vemanaeva ~]$ echo "test3" > /tmp/file01.txt
```

Рис. 26: Перезаписываем файл

Операцию выполнить удалось.

Проверим содержимое файла командой `cat /tmp/file01.txt`



```
[guest2@vemanaeva ~]$ cat /tmp/file01.txt  
test3  
[guest2@vemanaeva ~]$
```

Рис. 27: Смотрим содержимое файла

От пользователя guest2 попробуем удалить файл /tmp/file01.txt командой `rm /tmp/file01.txt`

```
test@vemanaeva:~$ rm /tmp/file01.txt  
rm: невозможно удалить «/tmp/file01.txt»: Операция не позволена
```

Рис. 28: Пытаемся удалить файл

Файл удалить не удалось.

Повысим свои права до суперпользователя следующей командой `su -` и выполним после этого команду, снимающую атрибут `t` (Sticky-бит) с директории `/tmp`, — `chmod -t /tmp`

```
[guest2@vemanaeva ~]$ su -  
Пароль:  
Последний вход в систему:Сб окт  7 14:02:53 MSK 2023на pts/1  
[root@vemanaeva ~]# chmod -t /tmp  
[root@vemanaeva ~]#
```

Рис. 29: Снимаем Sticky-бит с директории `/tmp`

```
[root@vemanaeva ~]# exit  
logout
```

Рис. 30: Logout

От пользователя guest2 проверим, что атрибута t у директории /tmp нет командой `ls -l / | grep tmp`

```
[guest2@vemanaeva ~]$ ls -l / | grep tmp
drwxrwxrwx. 23 root root 4096 окт  7 14:23 tmp
```

Рис. 31: Нет Sticky-бита среди атрибутов директории

Повторим шаги 4-9.

```
[guest2@vemanaeva ~]$ cat /tmp/file01.txt
test3
[guest2@vemanaeva ~]$ echo "test2" >> /tmp/file01.txt
[guest2@vemanaeva ~]$ cat /tmp/file01.txt
test3
test2
[guest2@vemanaeva ~]$ echo "test3" > /tmp/file01.txt
[guest2@vemanaeva ~]$ cat /tmp/file01.txt
test3
[guest2@vemanaeva ~]$ rm /tmp/file01.txt
[guest2@vemanaeva ~]$ ls -l /tmp/file01.txt
ls: невозможно получить доступ к /tmp/file01.txt: Нет такого файла или каталога
[guest2@vemanaeva ~]$
```

Рис. 32: Повторение. Получилось удалить файл, что не вышло в прошлый раз

В результате изменения удалось удалить файл от имени пользователя, не являющегося его владельцем.

Удалось удалить файл от имени пользователя, не являющегося его владельцем.

```
[guest2@vemanaeva ~]$ rm /tmp/file01.txt  
[guest2@vemanaeva ~]$ ls -l /tmp/file01.txt  
ls: невозможно получить доступ к /tmp/file01.txt: Нет такого файла или каталога
```

Рис. 33: Файл удалось удалить

Повысим свои права до суперпользователя и верните атрибут `t` на директорию `/tmp` командами:

```
[guest2@vemanaeva ~]$ su -  
Пароль:  
Последний вход в систему:Сб окт  7 14:22:37 MSK 2023на pts/1  
[root@vemanaeva ~]# chmod +t /tmp  
[root@vemanaeva ~]# exit  
logout
```

Рис. 34: Возвращаем атрибут директории

Выводы по проделанной работе

В результате выполнения работы мы изучили механизм идентификаторов, применения SetUID- и Sticky-битов, получили практические навыки работы в консоли с дополнительными атрибутами, рассмотрели работы механизма смены идентификатора процессов пользователей, а также посмотрели влияние бита Sticky на запись и удаление файлов.

Были записаны скринкасты выполнения и защиты лабораторной работы.