

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

Дисциплина: Информационная безопасность

Манаева Варвара Евгеньевна

Содержание

1	Техническое оснащение:	5
2	Цели и задачи работы	6
2.1	Цель	6
2.2	Задачи	6
3	Теоретическое введение [1]	7
4	Выполнение лабораторной работы [2]	8
4.1	Создание программ	8
4.2	Исследование Sticky-бита	17
5	Выводы по проделанной работе	22
5.1	Вывод	22
6	Список литературы	23

Список иллюстраций

4.1	Выполнен вход	8
4.2	Создана программа	9
4.3	Компилируем программу	9
4.4	Вывод программы simpleid	10
4.5	Вывод системной программы	10
4.6	Создание второй программы	11
4.7	Компиляция и запуск второй программы	11
4.8	Не получается выполнить команды	12
4.9	Меняем пользователя на суперпользователя и выполняем команды	12
4.10	Проверка	12
4.11	Выполняем программу и системную программу	13
4.12	Повторение действий с SetGID	13
4.13	Создаём программу	14
4.14	Компилируем программу и проверяем наличие	15
4.15	Изменяем права на файл	15
4.16	Проверка	15
4.17	Меняем владельца	16
4.18	Проверяем возможность чтения файла readfile.c	16
4.19	Проверяем возможность чтения файла /etc/shadow	17
4.20	Sticky-бит установлен на директории /tmp	17
4.21	Создадим файл	17
4.22	Выдаём права на запись категории пользователей “все остальные”	18
4.23	Смотрим содержимое файла	18
4.24	Дописываем информацию в файл	18
4.25	Смотрим содержимое файла	19
4.26	Перезаписываем файл	19
4.27	Смотрим содержимое файла	19
4.28	Пытаемся удалить файл	19
4.29	Снимаем Sticky-бит с директории /tmp	20
4.30	Logout	20
4.31	Нет Sticky-бита среди атрибутов директории	20
4.32	Повторение. Получилось удалить файл, что не вышло в прошлый раз	21
4.33	Файл удалось удалить	21
4.34	Возвращаем атрибут директории	21

Список таблиц

1 Техническое оснащение:

- Персональный компьютер с операционной системой Windows 10;
- Планшет для записи видеосопровождения и голосовых комментариев;
- Microsoft Teams, использующийся для записи скринкаста лабораторной работы;
- Приложение Rucharm для редактирования файлов формата *md*;
- *pandoc* для конвертации файлов отчётов и презентаций.

2 Цели и задачи работы

2.1 Цель

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.

2.2 Задачи

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

3 Теоретическое введение [1]

Setuid – это бит разрешения, который позволяет пользователю запускать исполняемый файл с правами владельца этого файла. Другими словами, использование этого бита позволяет нам поднять привилегии пользователя в случае, если это необходимо. Классический пример использования этого бита в операционной системе это команда `sudo`.

Принцип работы Setgid очень похож на setuid с отличием, что файл будет запускаться пользователем от имени группы, которая владеет файлом.

Последний специальный бит разрешения – это Sticky Bit. В случае, если этот бит установлен для папки, то файлы в этой папке могут быть удалены только их владельцем. Пример использования этого бита в операционной системе это системная папка `/tmp`. Эта папка разрешена на запись любому пользователю, но удалять файлы в ней могут только пользователи, являющиеся владельцами этих файлов.

4 Выполнение лабораторной работы [2]

4.1 Создание программ

1. Войдём в систему от имени пользователя guest. (4.1)

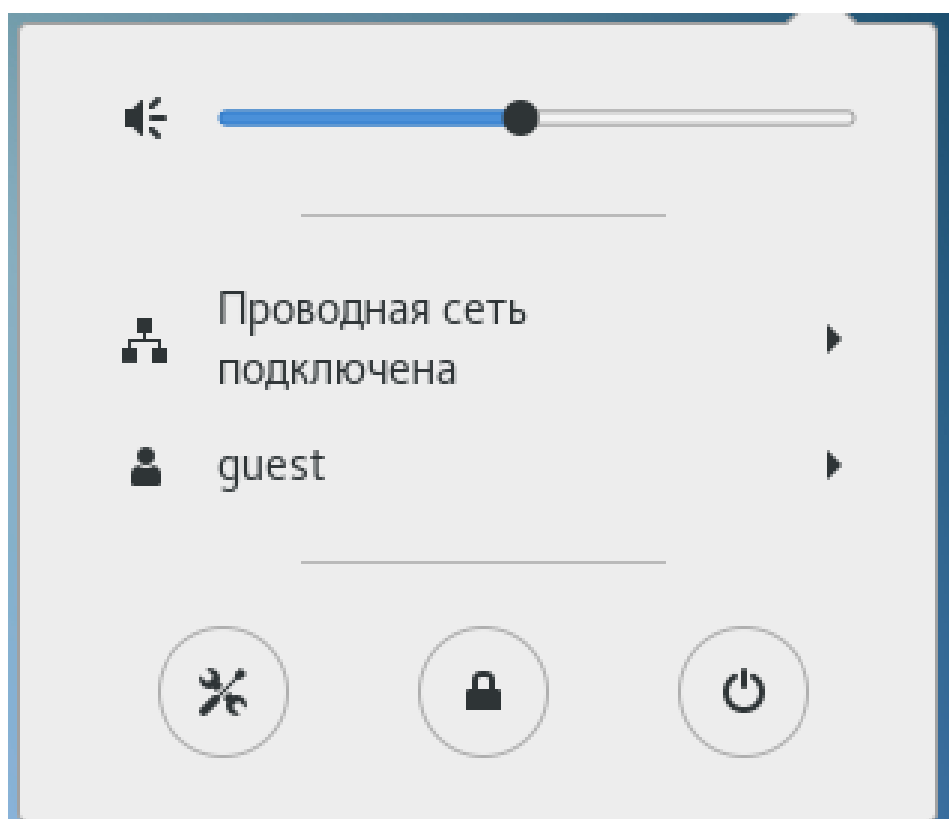


Рис. 4.1: Выполнен вход

2. Создадим программу simpleid.c (4.2):


```

#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;
}

```



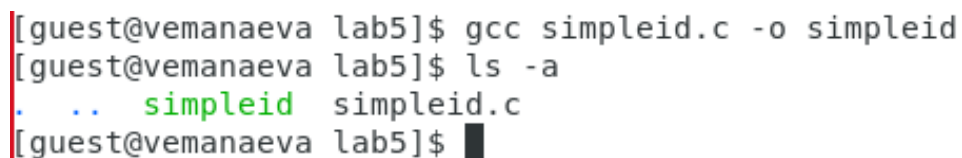
```

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. 4.2: Создана программа

3. Скомпилируем программу и убедимся, что файл программы создан, командой `gcc simpleid.c -o simpleid` (4.3)



```

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

```

Рис. 4.3: Компилируем программу

4. Выполним программу simpleid командой ./simpleid (4.4)

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

Рис. 4.4: Вывод программы simpleid

5. Выполним системную программу id id (4.5)

```
[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. 4.5: 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.

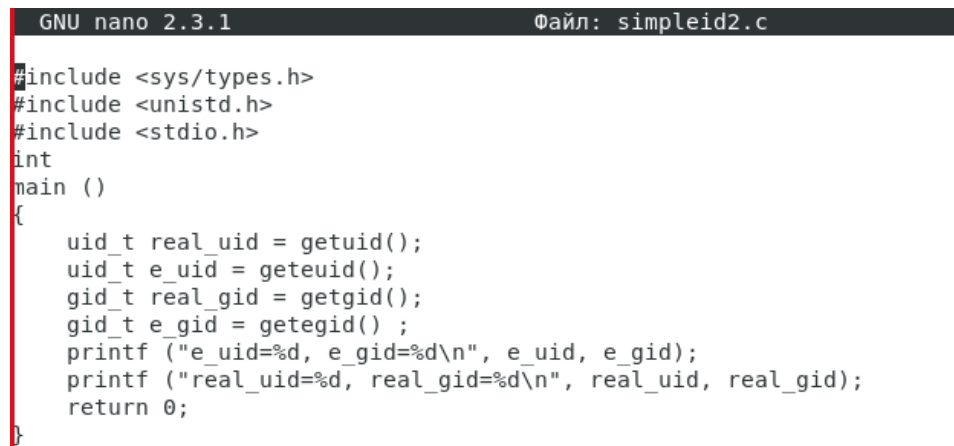
6. Усложним программу, добавив вывод действительных идентификаторов, и получившуюся программу назовём simpleid2.c (4.6).

```
#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;
}

```



```

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;
}

```

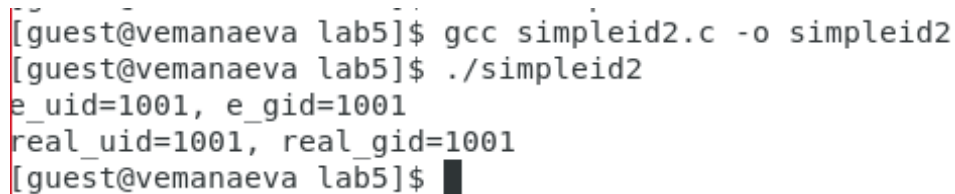
Рис. 4.6: Создание второй программы

7. Скомпилируем и запустим simpleid2.c командами (4.7):

```

gcc simpleid2.c -o simpleid2
./simpleid2

```



```

[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. 4.7: Компиляция и запуск второй программы

8. Попробуем выполнить команды (4.8):

```

chown root:guest /home/guest/lab5/simpleid2
chmod u+s /home/guest/lab5/simpleid2

```

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

Рис. 4.8: Не получается выполнить команды

9. Используйте `sudo` или повысьте временно свои права с помощью `su` и выполните эти команды. (4.9)

```
[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. 4.9: 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.

10. Выполним проверку правильности установки новых атрибутов и смены владельца файла `simpleid2` командой `ls -l simpleid2` (4.10)

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

Рис. 4.10: Проверка

11. Запустите `simpleid2` и `id` (4.11)

```
id
./simpleid2
```

```
[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) грппны=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 4.11: 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.

12. Прodelайте тоже самое относительно SetGID-бита (4.12).

```
chown root:root /home/guest/lab5/simpleid2
chmod g+s /home/guest/lab5/simpleid2
id
./simpleid2
```

```
[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) грппны=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. 4.12: Повторение действий с SetGID

13. Создайте программу `readfile.c` (4.13)

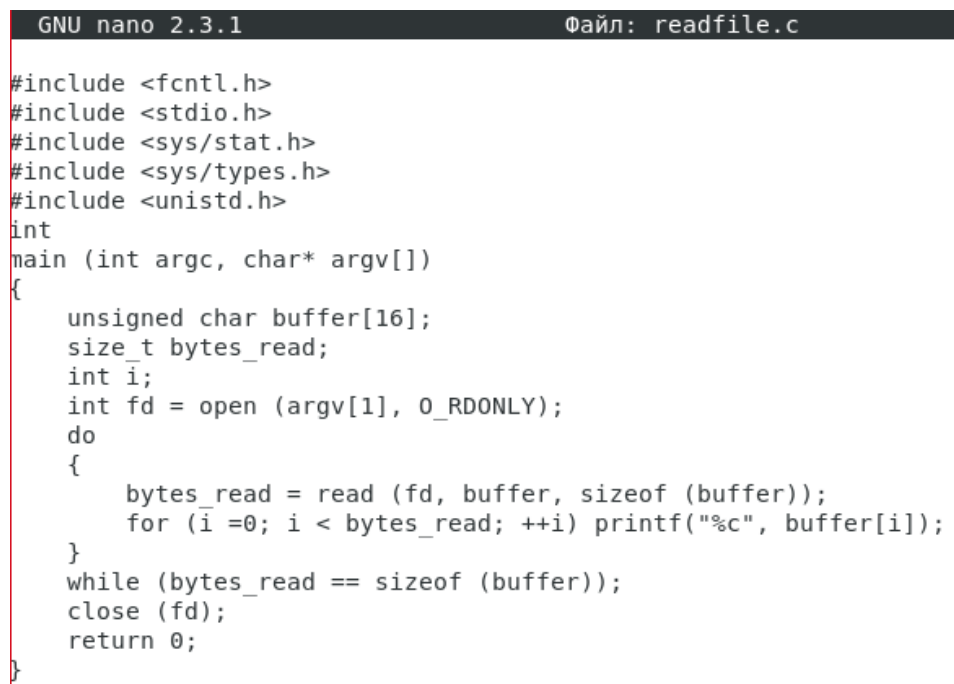
```
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

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;
}

```



```

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. 4.13: Создаём программу

14. Откомпилируйте её командой `gcc readfile.c -o readfile` (4.14)

```
[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. 4.14: Компилируем программу и проверяем наличие

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 (4.15).

```
chown root:root /home/guest/lab5/readfile.c
```

```
chmod 770 /home/guest/lab5/readfile.c
```

```
[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. 4.15: Изменяем права на файл

16. Проверьте, что пользователь `guest` не может прочитать файл `readfile.c` командой `cat /home/guest/lab5/readfile.c` (4.16)

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

Рис. 4.16: Проверка

17. Смените у программы `readfile` владельца и установите SetUID-бит (4.17).

```
chown root:root /home/guest/lab5/readfile
```

```
chmod u+s /home/guest/lab5/readfile
```

```
[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. 4.17: Меняем владельца

18. Проверьте, может ли программа readfile прочитать файл readfile.c (4.18)

```
[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. 4.18: Проверяем возможность чтения файла readfile.c

Да, программа может читать данный файл.

19. Проверьте, может ли программа readfile прочитать файл /etc/shadow (4.19)


```

[guest@vemanaeva lab5]$ ./readfile /etc/shadow
root:$6$5NDsC5pnY3mm94V7$0ek8eynNXXY1vHJA13rNFm8VE9gRctdwlCRS.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::::::
libstorageengine:!!:19609::::::
colord:!!:19609::::::
rpc:!!:19609:0:99999:7:::
kanned:!!:19609::::::

```

Рис. 4.19: Проверяем возможность чтения файла /etc/shadow

Да, программа может читать данный файл.

4.2 Исследование Sticky-бита

1. Выясним, установлен ли атрибут Sticky на директории /tmp командой `ls -l / | grep tmp` (4.20)

```

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

```

Рис. 4.20: Sticky-бит установлен на директории /tmp

2. От имени пользователя guest создадим файл file01.txt в директории /tmp со словом test командой `echo "test" > /tmp/file01.txt` (4.21)

```

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

```

Рис. 4.21: Создадим файл

3. Просмотрим атрибуты у только что созданного файла и разрешим чтение и запись для категории пользователей «все остальные» командами (4.22):

```
ls -l /tmp/file01.txt
chmod o+rw /tmp/file01.txt
ls -l /tmp/file01.txt
```

```
[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. 4.22: Выдаём права на запись категории пользователей “все остальные”

4. От пользователя guest2 (не являющегося владельцем) попробуем прочитать файл /tmp/file01.txt командой `cat /tmp/file01.txt` (4.23)

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

Рис. 4.23: Смотрим содержимое файла

5. От пользователя guest2 попробуем дозаписать в файл /tmp/file01.txt слово test2 командой `echo "test2" >> /tmp/file01.txt` (4.24)

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

Рис. 4.24: Дописываем информацию в файл

Операцию выполнить удалось.

6. Проверим содержимое файла командой `cat /tmp/file01.txt` (4.25)

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

Рис. 4.25: Смотрим содержимое файла

7. 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 (4.26)

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

Рис. 4.26: Перезаписываем файл

Операцию выполнить удалось.

8. Проверим содержимое файла командой cat /tmp/file01.txt (4.27)

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

Рис. 4.27: Смотрим содержимое файла

9. От пользователя guest2 попробуем удалить файл /tmp/file01.txt командой rm /tmp/file01.txt (4.28)

```
[guest2@vemanaeva ~]$ rm /tmp/file01.txt
rm: невозможно удалить «/tmp/file01.txt»: Операция не позволена
```

Рис. 4.28: Пытаемся удалить файл

Файл удалить не удалось.

10. Повысим свои права до суперпользователя следующей командой `su -` и выполним после этого команду, снимающую атрибут `t` (Sticky-бит) с директории `/tmp`, — `chmod -t /tmp` (4.29)

```
[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. 4.29: Снимаем Sticky-бит с директории `/tmp`

11. Покинем режим суперпользователя командой `exit` (4.30)

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

Рис. 4.30: Logout

12. От пользователя `guest2` проверим, что атрибута `t` у директории `/tmp` нет командой `ls -l / | grep tmp` (4.31)

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

Рис. 4.31: Нет Sticky-бита среди атрибутов директории

13. Повторим шаги 4-9. (4.32)

```
[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. 4.32: Повторение. Получилось удалить файл, что не вышло в прошлый раз

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

14. Удалось удалить файл от имени пользователя, не являющегося его владельцем (4.33).

```
[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. 4.33: Файл удалось удалить

15. Повысим свои права до суперпользователя и верните атрибут t на директорию /tmp командами (4.34):

```
su -
```

```
chmod +t /tmp
```

```
exit
```

```
[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. 4.34: Возвращаем атрибут директории

5 Выводы по проделанной работе

5.1 Вывод

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.

Ссылки на скринкасты:

- Выполнение, Youtube
- Выполнение, Rutube
- Защита презентации, Youtube
- Защита презентации, Rutube

6 Список литературы

1. Колисниченко Д. Linux. От новичка к профессионалу. В подлиннике. 8-е изд. 2022. 688 с.
2. Лабораторная работа № 5. Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов [Электронный ресурс]. Российский Университет Дружбы Народов имени Патрису Лумумбы, 2023. URL: <https://esystem.rudn.ru/mod/resource/view.php?id=1031377>.