

Лабораторная работа №5

Дисциплина: Информационная безопасность

Дорофеева Алёна Тимофеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Создание программы	9
4.2	Исследование Sticky-бита	19
5	Выводы	24
	Список литературы	25

Список иллюстраций

4.1	Проверка необходимых ресурсов	9
4.2	Создание и компиляция файла	10
4.3	Программа simpleid.c	11
4.4	Вызов программы simpleid.c	11
4.5	Создание и компиляция simpleid2.c	13
4.6	Добавление SETUID	13
4.7	Проверка добавления SETUID-бита	14
4.8	Выполнение программы после добавления SETUID-бита	14
4.9	Добавление setGID-бита	15
4.10	Выполнение программы после добавления SETUID-бита	15
4.11	Создание и компиляция программы readfile.c	16
4.12	Программа readfile.c	17
4.13	Проверка запрета на чтение файла	17
4.14	Назначение SetUID-бита	17
4.15	Попытка прочитать файл readfile.c через программу readfile	18
4.16	Попытка прочитать файл /etc/shadow через программу readfile	19
4.17	Проверка наличия Sticky-бита	19
4.18	Создание и запись в файл file01.txt	20
4.19	guest2 - чтение файла	20
4.20	Попытка удаления файла со Sticky-битом	21
4.21	Снятие с директории Sticky-бита	22
4.22	Проверка снятия с директории Sticky-бита	22
4.23	Попытка удаления файла без Sticky-бита	22
4.24	Возвращение Sticky-бита	23

Список таблиц

1 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.

2 Задание

- Изучить на практике применение SetUID- и Sticky-битов.

3 Теоретическое введение

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`.

Там, где обычно установлен классический бит `x` (на исполнение), выставляется специальный бит `s`. Это позволяет обычному пользователю системы выполнять команды с повышенными привилегиями без необходимости входа в систему как `root`, разумеется зная пароль пользователя `root`. Принцип работы **Setgid** очень похож на `setuid` с отличием, что файл будет запускаться пользователем от имени группы, которая владеет файлом.

Биты **setuid** и **setgid** выставляются с помощью команд `chmod u+s` и `chmod g+s` соответственно.[01?]

Sticky-bit — дополнительный атрибут файлов или каталогов в операционных системах семейства UNIX. В настоящее время `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. Установка атрибута производится утилитой `chmod`.

В операционной системе Solaris для файлов, не являющихся программами, имеет строго противоположное действие — запрещает сохранение данных этого

файла в системном кэше.[02?]

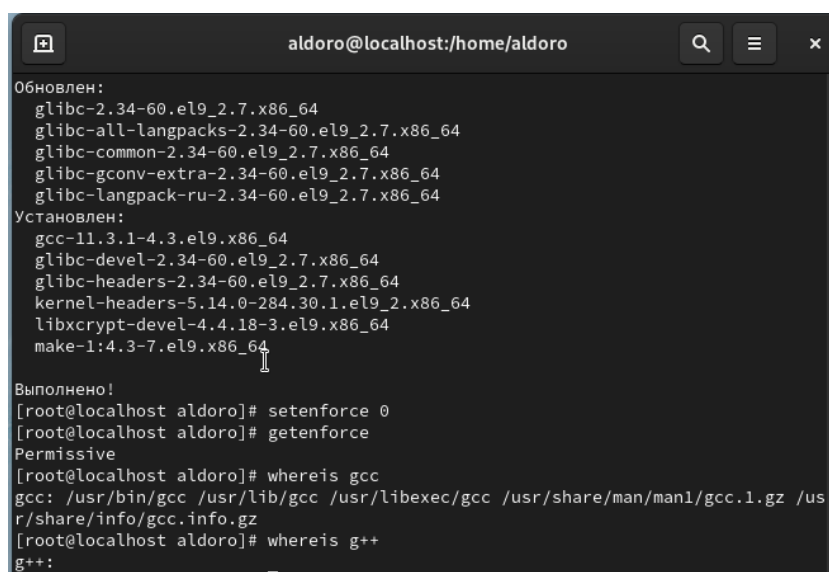
Это разрешение полезно для защиты файлов от случайного удаления в среде, где несколько пользователей имеют права на запись в один и тот же каталог. Если применяется закрепленный sticky bit, пользователь может удалить файл, только если он является пользователем-владельцем файла или каталога, в котором содержится файл. По этой причине он применяется в качестве разрешения по умолчанию для каталога /tmp и может быть полезен также для каталогов общих групп.

Когда вы применяете sticky bit, пользователь может удалять файлы, только если выполняется одно из следующих условий:

- Пользователь является владельцем файла;
- Пользователь является владельцем каталога, в котором находится файл.[03?]

4 Выполнение лабораторной работы

Прежде чем начать выполнять работу, проверим, имеются ли на устройстве все необходимые ресурсы для осуществления компиляции (рис. 4.1).



```
aldoro@localhost:/home/aldoro
Обновлен:
glibc-2.34-60.el9_2.7.x86_64
glibc-all-langpacks-2.34-60.el9_2.7.x86_64
glibc-common-2.34-60.el9_2.7.x86_64
glibc-gconv-extra-2.34-60.el9_2.7.x86_64
glibc-langpack-ru-2.34-60.el9_2.7.x86_64
Установлен:
gcc-11.3.1-4.3.el9.x86_64
glibc-devel-2.34-60.el9_2.7.x86_64
glibc-headers-2.34-60.el9_2.7.x86_64
kernel-headers-5.14.0-284.30.1.el9_2.x86_64
libxcrypt-devel-4.4.18-3.el9.x86_64
make-1:4.3-7.el9.x86_64
Выполнено!
[root@localhost aldoro]# setenforce 0
[root@localhost aldoro]# getenforce
Permissive
[root@localhost aldoro]# whereis gcc
gcc: /usr/bin/gcc /usr/lib/gcc /usr/libexec/gcc /usr/share/man/man1/gcc.1.gz /usr/share/info/gcc.info.gz
[root@localhost aldoro]# whereis g++
g++:
```

Рис. 4.1: Проверка необходимых ресурсов

4.1 Создание программы

1. Войдите в систему от имени пользователя guest.
2. Создайте программу simpleid.c (рис. 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;
}

```

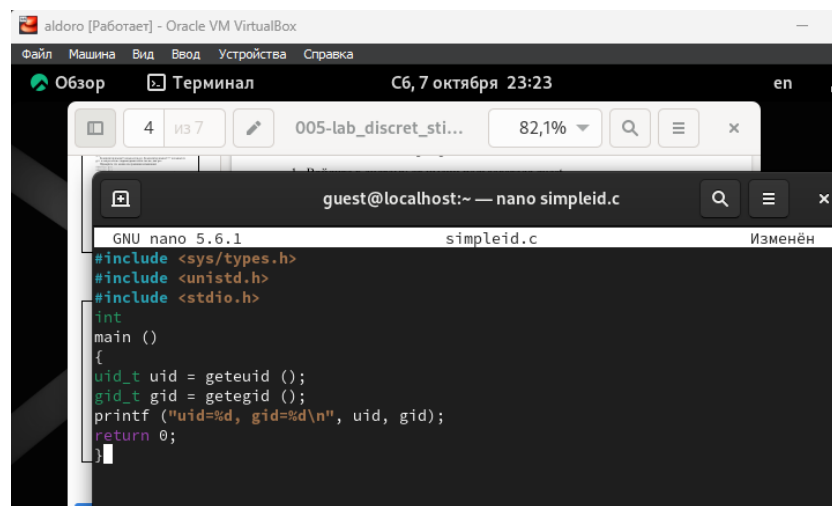


Рис. 4.2: Создание и компиляция файла

```

[guest@aldoro ~]$ touch simpleid
[guest@aldoro ~]$ nano simpleid
[guest@aldoro ~]$ cat simpleid
#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;
}
[guest@aldoro ~]$ gcc simpleid.c -o simpleid
cc1: fatal error: simpleid.c: No such file or directory
compilation terminated.
[guest@aldoro ~]$ mv simpleid simpleid.c
[guest@aldoro ~]$ gcc simpleid.c -o simpleid
[guest@aldoro ~]$ gcc -c simpleid.c
[guest@aldoro ~]$ ls
Desktop  Documents  Music      Public     simpleid.c  Templates
dir1     Downloads  Pictures   simpleid   simpleid.o  Videos
[guest@aldoro ~]$

```

Рис. 4.3: Программа simpleid.c

3. Скомпилируйте программу и убедитесь, что файл программы создан: `gcc simpleid.c -o simpleid` (рис. 4.3).

Видим, что никаких ошибок при компиляции не возникает, файлы успешно созданы (проверка через команду `ls`).

4. Выполните программу `simpleid` (рис. 4.4): `./simpleid`.

```

[guest@aldoro ~]$ ./simpleid
uid=1001, gid=1001
[guest@aldoro ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@aldoro ~]$

```

Рис. 4.4: Вызов программы simpleid.c

5. Выполните системную программу `id` (рис. 4.4): `id`. и сравните полученный вами результат с данными предыдущего пункта задания.

Видим, что выводы программы, написанной ранее, и команды `id` совпадают - в каждой из них предоставляется информация об `id` владельца и группы, ими является `guest` с `uid = 1001`, `gid = 1001`.

6. Усложните программу, добавив вывод действительных идентификаторов.

Скомпилируйте и запустите `simpleid2.c` (рис. ??):(рис. 4.5):

```
#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;
}
```

Получившуюся программу назовите `simpleid2.c` (рис. 4.5).

```

[guest@aldoro ~]$ nano simpleid2.c
[guest@aldoro ~]$ cat 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;
}
[guest@aldoro ~]$ ls
Desktop  Documents  Music      Public    simpleid2.c  simpleid.o  Videos
dir1     Downloads  Pictures   simpleid  simpleid.c   Templates
[guest@aldoro ~]$ gcc simpleid2.c -o simpleid2
[guest@aldoro ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@aldoro ~]$

```

Рис. 4.5: Создание и компиляция simpleid2.c

Видим, что теперь выводятся айди владельца файла и айди текущего пользователя - сейчас это guest с uid = 1001, gid = 1001.

8. От имени суперпользователя выполните команды (рис. 4.6):

```
chown root:guest /home/guest/simpleid2
```

```
chmod u+s /home/guest/simpleid2
```

```

[guest@aldoro ~]$ su aldoro
Password:
[aldoro@aldoro guest]$ sudo -i
[sudo] password for aldoro:
[root@aldoro ~]# chown root:guest /home/guest/simpleid2
[root@aldoro ~]# chmod u+s /home/guest/simpleid2
[root@aldoro ~]#

```

Рис. 4.6: Добавление SETUID

9. Используйте sudo или повысьте временно свои права с помощью su. Поясните, что делают эти команды.

Команды отличаются следующим: *su* требует пароль целевой учетной записи (например, пользователя *root*) и переключает вас на нее, в то время как *sudo* требует пароль текущего пользователя и запускает от его имени только лишь одну (или несколько) команд, на выполнение которых требуются права суперпользователя.

10. Выполните проверку правильности установки новых атрибутов и смены владельца файла *simpleid2* (рис. 4.7): `ls -l simpleid2`

```
[aldoro@aldoro guest]$ su guest
Password:
[guest@aldoro ~]$ ls -l simpleid2
-rwsr-xr-x. 1 root guest 26064 Oct 12 05:09 simpleid2
[guest@aldoro ~]$
```

Рис. 4.7: Проверка добавления SETUID-бита

Видим добавленный SetUID-бит с правах доступа пользователя-владельца - **s**.

11. Запустите *simpleid2* и *id*:

`./simpleid2`

`id`

```
[guest@aldoro ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@aldoro ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@aldoro ~]$
```

Рис. 4.8: Выполнение программы после добавления SETUID-бита

Видим, что теперь айди пользователя владельца - это айди пользователя *root*.

12. Прodelайте тоже самое относительно SetGID-бита.

```

[guest@aldoro ~]$ su aldoro
Password:
[aldoro@aldoro guest]$ sudo -i
[root@aldoro ~]# chown root:root /home/guest/simpleid2
[root@aldoro ~]# chmod g+s /home/guest/simpleid2
[root@aldoro ~]#

```

Рис. 4.9: Добавление setGID-бита

От пользователя guest смотрим назначение бита для группы - видим, что он добавился (рис. 4.10). Запускаем программу и видим теперь поменявшееся айди группы пользователей на айди root (рис. 4.10).

```

[aldoro@aldoro guest]$ su guest
Password:
[guest@aldoro ~]$ ls -l simpleid2
-rwxr-sr-x. 1 root root 26064 Oct 12 05:09 simpleid2
[guest@aldoro ~]$ ./simpleid2
e_uid=1001, e_gid=0
real_uid=1001, real_gid=1001
[guest@aldoro ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@aldoro ~]$

```

Рис. 4.10: Выполнение программы после добавления SETUID-бита

13. Создайте программу readfile.c. Откомпилируйте её (рис. ??):(рис. ??):

```

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

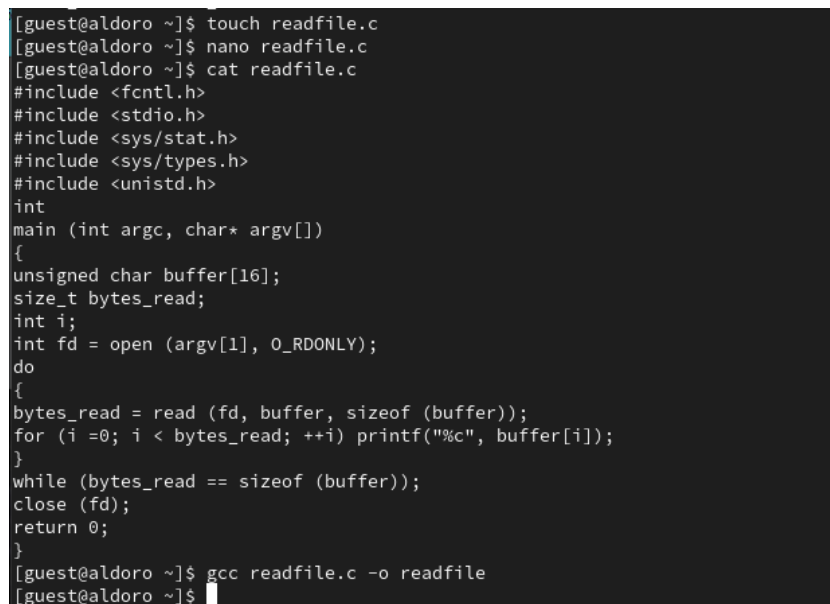
int
main (int argc, char* argv[])
{
    unsigned char buffer[16];
    size_t bytes_read;
    int i;

```

```

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@aldoro ~]$ touch readfile.c
[guest@aldoro ~]$ nano readfile.c
[guest@aldoro ~]$ cat 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@aldoro ~]$ gcc readfile.c -o readfile
[guest@aldoro ~]$

```

Рис. 4.11: Создание и компиляция программы readfile.c

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.

Владельца файла меняем на root, назначаем права доступа 733 - дают полные права пользователю-владельцу и права на запись и выполнение остальным

пользователям и группе (рис. 4.12).

```
[guest@aldoro ~]$ su aldoro
Password:
[aldoro@aldoro guest]$ sudo -i
[sudo] password for aldoro:
[root@aldoro ~]# chown root:guest /home/guest/readfile.c
[root@aldoro ~]# chmod 733 /home/guest/readfile.c
[root@aldoro ~]#
```

Рис. 4.12: Программа readfile.c

16. Проверьте, что пользователь guest не может прочитать файл readfile.c (рис. 4.13).

```
[aldoro@aldoro guest]$ su guest
Password:
[guest@aldoro ~]$ ls -l readfile.c
-rwx-wx-wx. 1 root guest 402 Oct 12 05:19 readfile.c
[guest@aldoro ~]$ cat readfile.c
cat: readfile.c: Permission denied
[guest@aldoro ~]$
```

Рис. 4.13: Проверка запрета на чтение файла

Пользователь действительно не может прочитать файл readfile.c - выводится сообщение **Permission denied**.

17. Смените у программы readfile владельца и установите SetUID-бит.

Меняем владельца на root и назначаем SetUID-бит (рис. 4.14).

```
[aldoro@aldoro guest]$ sudo -i
[root@aldoro ~]# chown root:guest /home/guest/readfile
[root@aldoro ~]# chmod u+s /home/guest/readfile
[root@aldoro ~]#
```

Рис. 4.14: Назначение SetUID-бита

18. Проверьте, может ли программа readfile прочитать файл readfile.c (рис. 4.15)?

```
[guest@aldoro ~]$ ./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@aldoro ~]$
```

Рис. 4.15: Попытка прочитать файл readfile.c через программу readfile

Нам удалось прочитать файл.

19. Проверьте, может ли программа readfile прочитать файл /etc/shadow? Отрадите полученный результат и ваши объяснения в отчёте.

2. От имени пользователя guest создайте файл file01.txt в директории /tmp со словом test: `echo "test" > /tmp/file01.txt` (рис. 4.18)

```
[guest@aldoro ~]$ echo "test" > /tmp/file01.txt
[guest@aldoro ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 Oct 12 05:30 /tmp/file01.txt
[guest@aldoro ~]$ chmod o+rw /tmp/file01.txt
[guest@aldoro ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 Oct 12 05:30 /tmp/file01.txt
[guest@aldoro ~]$
```

Рис. 4.18: Создание и запись в файл file01.txt

3. Просмотрите атрибуты у только что созданного файла и разрешите чтение и запись для категории пользователей «все остальные» (рис. 4.18):

```
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` (рис. 4.19).

```
[guest@aldoro ~]$ su guest2
Password:
[guest2@aldoro guest]$ cat /tmp/file01.txt
test
[guest2@aldoro guest]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@aldoro guest]$ cat /tmp/file01.txt
test
[guest2@aldoro guest]$ echo "test3" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@aldoro guest]$ cat /tmp/file01.txt
test
[guest2@aldoro guest]$
```

Рис. 4.19: guest2 - чтение файла

Содержимое файла выводится - можем прочитать файл.

5. От пользователя guest2 попробуйте дозаписать в файл /tmp/file01.txt слово test2 командой `echo "test2" >> /tmp/file01.txt` (рис. 4.19). Удалось ли вам выполнить операцию?

Выполнить операцию не удалось - Permission denied.

6. Проверьте содержимое файла командой `cat /tmp/file01.txt` (рис. 4.19).

Файл никак не изменился.

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` (рис. 4.19). Удалось ли вам выполнить операцию?

Выполнить операцию не удалось - Permission denied.

8. Проверьте содержимое файла командой `cat /tmp/file01.txt` (рис. 4.19).

Файл никак не изменился.

9. От пользователя guest2 попробуйте удалить файл /tmp/file01.txt командой `rm /tmp/file01.txt` (рис. 4.20). Удалось ли вам удалить файл?



```
[guest2@aldoro guest]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest2@aldoro guest]$
```

Рис. 4.20: Попытка удаления файла со Sticky-битом

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` (рис. 4.21).

```
[guest2@aldoro guest]$ su -
Password:
[root@aldoro ~]# chmod -t /tmp
[root@aldoro ~]# exit
logout
[guest2@aldoro guest]$
```

Рис. 4.21: Снятие с директории Sticky-бита

11. Покиньте режим суперпользователя командой `exit` (рис. 4.21).
12. От пользователя `guest2` проверьте, что атрибута `t` у директории `/tmp` нет:
`ls -l / | grep tmp` (рис. 4.22)

```
[guest2@aldoro guest]$ ls -l / | grep tmp
drwxrwxrwx. 18 root root 4096 Oct 12 05:35 tmp
[guest2@aldoro guest]$ cat /tmp/file01.txt
test
[guest2@aldoro guest]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@aldoro guest]$ echo "test3" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@aldoro guest]$ cat /tmp/file01.txt
test
[guest2@aldoro guest]$
```

Рис. 4.22: Проверка снятия с директории Sticky-бита

13. Повторите предыдущие шаги (рис. 4.22). Какие наблюдаются изменения?

Изменений не наблюдается, мы также можем читать файл, но не можем производить дозапись и запись в него.

14. Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем (рис. 4.23)? Ваши наблюдения занесите в отчёт.

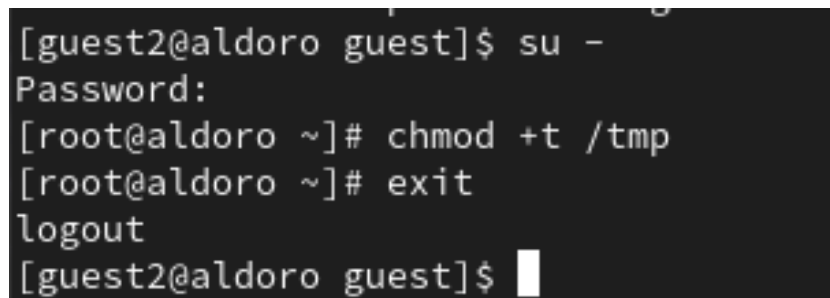
```
[guest2@aldoro guest]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
[guest2@aldoro guest]$
```

Рис. 4.23: Попытка удаления файла без Sticky-бита

В этот раз без Sticky-бита нам удалось успешно удалить файл file01.txt.

15. Повысьте свои права до суперпользователя и верните атрибут t на директорию /tmp (рис. 4.24):

```
su -  
chmod +t /tmp  
exit
```



```
[guest2@aldoro guest]$ su -  
Password:  
[root@aldoro ~]# chmod +t /tmp  
[root@aldoro ~]# exit  
logout  
[guest2@aldoro guest]$
```

Рис. 4.24: Возвращение Sticky-бита

5 Выводы

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

Список литературы