

Лабораторная работа №5

Дисциплина: Основы информационной безопасности

Дарижапов Тимур Андреевич

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выводы	15
5	Список литературы	16

List of Figures

3.1	Предварительная подготовка	7
3.2	Команда “whereis”	7
3.3	Вход в систему и создание программы	8
3.4	Код программы simpleid.c	8
3.5	Компиляция и выполнение программы simpleid	9
3.6	Усложнение программы	9
3.7	Компиляция и выполнение программы simpleid2	9
3.8	Установка новых атрибутов (SetUID) и смена владельца файла . .	10
3.9	Запуск simpleid2 после установки SetUID	10
3.10	Запуск simpleid2 после установки SetGID	10
3.11	Код программы readfile.c	11
3.12	Смена владельца и прав доступа у файла readfile.c	11
3.13	Запуск программы readfile	12
3.14	Создание файла file01.txt	12
3.15	Попытка выполнить действия над файлом file01.txt от имени поль- зователя guest2	13
3.16	Удаление атрибута t (Sticky-бита) и повторение действий	14
3.17	Возвращение атрибута t (Sticky-бита)	14

List of Tables

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 Теоретическое введение

SetUID, SetGID и Sticky - это специальные типы разрешений позволяют задавать расширенные права доступа на файлы или каталоги. • SetUID (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. • SetGID (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. • Sticky bit в основном используется в общих каталогах, таких как /var или /tmp, поскольку пользователи могут создавать файлы, читать и выполнять их, принадлежащие другим пользователям, но не могут удалять файлы, принадлежащие другим пользователям. Более подробно см. в [1].

3 Выполнение лабораторной работы

1 часть: Создание программы

1)Для начала мы убеждаемся, что компилятор gcc установлен, используя команду “gcc -v”. Затем отключаем систему запретов до очередной перезагрузки системы командой “sudo setenforce 0”, после чего команда “getenforce” выводит “Permissive”(Рисунок 3.1).

```
[tadarizhapov@tadarizhapov ~]$ gcc -v
Используются внутренние спецификации.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/11/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Целевая архитектура: x86_64-redhat-linux
Параметры конфигурации: ../configure --enable-bootstrap --enable-host-pie --enable-host-bind-now
--enable-languages=c,c++,fortran,lto --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share
/info --with-bugurl=https://bugs.rockylinux.org/ --enable-shared --enable-threads=posix --enable
-checking=release --with-system-zlib --enable-_cxa_atexit --disable-libunwind-exceptions --enab
le-gnu-unique-object --enable-linker-build-id --with-gcc-major-version-only --enable-plugin --en
able-initfini-array --without-isl --enable-multilib --with-linker-hash-style=gnu --enable-offloa
d-targets=nvptx-none --without-cuda-driver --enable-gnu-indirect-function --enable-cet --with-tu
ne=generic --with-arch_64=x86-64-v2 --with-arch_32=x86-64 --build=x86_64-redhat-linux --with-bui
ld-config=bootstrap-lto --enable-link-serialization=1
Модель многопоточности: posix
Supported LTO compression algorithms: zlib zstd
gcc версия 11.3.1 20211121 (Red Hat 11.3.1-4) (GCC)
[tadarizhapov@tadarizhapov ~]$ sudo setenforce 0
[sudo] пароль для tadarizhapov:
[tadarizhapov@tadarizhapov ~]$ getenforce
Permissive
[tadarizhapov@tadarizhapov ~]$
```

Figure 3.1: Предварительная подготовка

2)Проверяем успешное выполнение команд “whereis gcc” и “whereis g++”(их расположение)(Рисунок 3.2).

```
[tadarizhapov@tadarizhapov ~]$ 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
[tadarizhapov@tadarizhapov ~]$ whereis g++
g++: /usr/bin/g++ /usr/share/man/man1/g++.1.gz
[tadarizhapov@tadarizhapov ~]$
```

Figure 3.2: Команда “whereis”

3)Входим в систему от имени пользователя guest командой “su - guest”. Создаём программу simpleid.c командой “touch simpleid.c” и открываем её в редакторе командой “gedit /home/guest/lab05/simpleid.c” (Рисунок 3.3).

```
[tadarizhapov@tadarizhapov ~]$ su - guest
Пароль:
[guest@tadarizhapov ~]$ ls
dir1  Документы  Изображения  Общедоступные  Рабочий стол
[guest@tadarizhapov ~]$ mkdir lab05
[guest@tadarizhapov ~]$ cd lab05
[guest@tadarizhapov lab05]$ touch simpleid.c
[guest@tadarizhapov lab05]$ ls
simpleid.c
[guest@tadarizhapov lab05]$ gedit /home/guest/lab05/simpleid.c

(gedit:3317): dbind-WARNING **: 20:15:19.382: Couldn't register with accessibility bus: Did not receive a reply. Possible causes include: the remote application did not send a reply, the message bus security policy blocked the reply, the reply timeout expired, or the network connection was broken.

(gedit:3317): dconf-WARNING **: 20:15:19.417: failed to commit changes to dconf: Не удалось выполнить процесс-потомок «dbus-launch» (Нет такого файла или каталога)

(gedit:3317): dconf-WARNING **: 20:15:19.428: failed to commit changes to dconf: Не удалось выполнить процесс-потомок «dbus-launch» (Нет такого файла или каталога)

(gedit:3317): dconf-WARNING **: 20:15:19.612: failed to commit changes to dconf: Не удалось выполнить процесс-потомок «dbus-launch» (Нет такого файла или каталога)

(gedit:3317): dconf-WARNING **: 20:15:19.613: failed to commit changes to dconf: Не удалось выполнить процесс-потомок «dbus-launch» (Нет такого файла или каталога)

(gedit:3317): dconf-WARNING **: 20:15:19.613: failed to commit changes to dconf: Не удалось выполнить процесс-потомок «dbus-launch» (Нет такого файла или каталога)
```

Figure 3.3: Вход в систему и создание программы

4)Код программы выглядит следующим образом (Рисунок 3.4).

```
*simpleid.c
~/lab05

1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4
5 int
6 main ()
7 {
8     uid_t uid = geteuid ();
9     gid_t gid = getegid ();
10    printf("uid=%d, gid=%d\n", uid, gid);
11    return 0;
12 }
```

Figure 3.4: Код программы simpleid.c

5)Скомпилируем программу и убедимся, что файл программы был создан командой “gcc simpleid.c -o simpleid”. Выполняем программу simpleid командой “./simpleid”, а затем системную программу id командой “id”. Результаты, полученные в результате выполнения обеих команд, совпадают(uid=1001 и gid=1001) (Рисунок 3.5).


```
[guest@tadarizhapov lab05]$ gcc simpleid.c -o simpleid
[guest@tadarizhapov lab05]$ ./simpleid
uid=1001, gid=1001
[guest@tadarizhapov lab05]$ id
uid=1001(guest) gid=1001(guest) rгруппы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@tadarizhapov lab05]$
```

Figure 3.5: Компиляция и выполнение программы simpleid

6) Усложняем программу, добавив вывод действительных идентификаторов, новый файл назовём simpleid.c (Рисунок 3.6).

```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4
5 int
6 main ()
7 {
8     uid_t real_uid = getuid ();
9     uid_t e_uid = geteuid ();
10
11     gid_t real_gid = getgid ();
12     gid_t e_gid = getegid ();
13
14     printf("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
15     printf("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
16     return 0;
17 }
```

Figure 3.6: Усложнение программы

7) Скомпилируем и запустим simpleid2.c командами “gcc simpleid2.c -o simpleid2” и “./simpleid2” (Рисунок 3.7).

```
[guest@tadarizhapov lab05]$ gcc simpleid2.c -o simpleid2
[guest@tadarizhapov lab05]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@tadarizhapov lab05]$
```

Figure 3.7: Компиляция и выполнение программы simpleid2

8) От имени суперпользователя выполняем команды “sudo chown root:guest /home/guest/lab05/simpleid2” и “sudo chmod u+s /home/guest/lab05/simpleid2”, затем выполняем проверку правильности установки новых атрибутов и смены владельца файла simpleid2 командой “sudo ls -l /home/guest/lab05/simpleid2” (Рисунок 3.8). Этими командами была произведена смена пользователя файла на root и установлен SetUID-бит.

```
tadarizhapov@tadarizhapov:~  
[tadarizhapov@tadarizhapov ~]$ sudo chown root:guest /home/guest/lab05/simpleid2  
[sudo] пароль для tadarizhapov:  
[tadarizhapov@tadarizhapov ~]$ sudo chmod u+s /home/guest/lab05/simpleid2  
[tadarizhapov@tadarizhapov ~]$ sudo ls -l /home/guest/lab05/simpleid2  
-rwsr-xr-x. 1 root guest 26064 окт  3 22:12 /home/guest/lab05/simpleid2  
[tadarizhapov@tadarizhapov ~]$
```

Figure 3.8: Установка новых атрибутов (SetUID) и смена владельца файла

9)Запускаем программы simpleid2 и id. Теперь появились различия в uid (Рисунок 3.9).

```
[guest@tadarizhapov lab05]$ ./simpleid2  
e_uid=0, e_gid=1001  
real_uid=1001, real_gid=1001  
[guest@tadarizhapov lab05]$ id  
uid=1001(guest) gid=1001(guest) rpyнны=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
[guest@tadarizhapov lab05]$
```

Figure 3.9: Запуск simpleid2 после установки SetUID

10)Прделаем тоже самое относительно SetGID-бита. Также можем заметить различия с предыдущим пунктом (Рисунок 3.10).

```
[tadarizhapov@tadarizhapov ~]$ sudo chown root:guest /home/guest/lab05/simpleid2  
[sudo] пароль для tadarizhapov:  
[tadarizhapov@tadarizhapov ~]$ sudo chmod g+s /home/guest/lab05/simpleid2  
[tadarizhapov@tadarizhapov ~]$ sudo ls -l /home/guest/lab05/simpleid2  
-rwxr-sr-x. 1 root guest 26064 окт  3 22:12 /home/guest/lab05/simpleid2  
[tadarizhapov@tadarizhapov ~]$  
  
guest@tadarizhapov:~/lab05  
[guest@tadarizhapov lab05]$ ./simpleid2  
e_uid=1001, e_gid=1001  
real_uid=1001, real_gid=1001  
[guest@tadarizhapov lab05]$ id  
uid=1001(guest) gid=1001(guest) rpyнны=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
[guest@tadarizhapov lab05]$
```

Figure 3.10: Запуск simpleid2 после установки SetGID

11)Создаем программу readfile.c(Рисунок 3.11).

```

1 #include <fcntl.h>
2 #include <stdio.h>
3 #include <sys/stat.h>
4 #include <sys/types.h>
5 #include <unistd.h>
6
7 int
8 main (int argc, char* argv[])
9 {
10     unsigned char buffer[16];
11     size_t bytes_read;
12     int i;
13     int fd = open (argv[1], O_RDONLY);
14     do
15     {
16         bytes_read = read (fd, buffer, sizeof (buffer));
17         for (i = 0; i < bytes_read; ++i) printf("%c", buffer[i]);
18     }
19
20     while (bytes_read == sizeof (buffer));
21     close (fd);
22     return 0;
23 }

```

Figure 3.11: Код программы readfile.c

12)Скомпилируем созданную программу командой “gcc readfile.c -o readfile”. Сменим владельца у файла readfile.c командой “sudo chown root:guest /home/guest/readfile.c” и поменяем права так, чтобы только суперпользователь мог прочитать его, а guest не мог, с помощью команды “sudo chmod 700 /home/guest/readfile.c”. Убеждаемся, что пользователь guest не может прочитать файл readfile.c командой “cat readfile.c”, получив отказ в доступе (Рисунок 3.12).

```

guest@tadarizhapov:~/lab05
[guest@tadarizhapov lab05]$ gcc readfile.c -o readfile
[guest@tadarizhapov lab05]$ ls
readfile readfile.c simpleid simpleid2 simpleid2.c simpleid.c
[guest@tadarizhapov lab05]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@tadarizhapov lab05]$

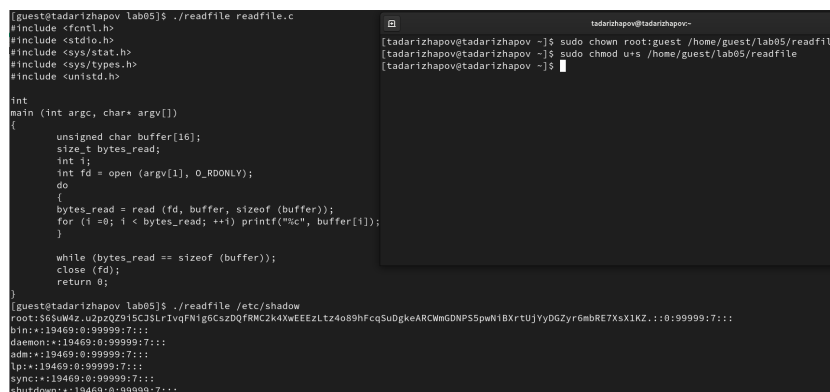
tadarizhapov@tadarizhapov:~
[tadarizhapov@tadarizhapov ~]$ sudo chown root:guest /home/guest/lab05/readfile.c
[tadarizhapov@tadarizhapov ~]$ sudo chmod 700 /home/guest/lab05/readfile.c
[tadarizhapov@tadarizhapov ~]$

```

Figure 3.12: Смена владельца и прав доступа у файла readfile.c

13)Поменяем владельца у программы readfile и установим SetUID. Проверим, может ли программа readfile прочитать файл readfile.c командой “./readfile readfile.c”. Прочитать удалось. Аналогично проверяем, можно ли прочитать файл

/etc/shadow. Прочитать удалось (Рисунок 3.13).



```
guest@tadarizhapov lab05$ ./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]);
    }

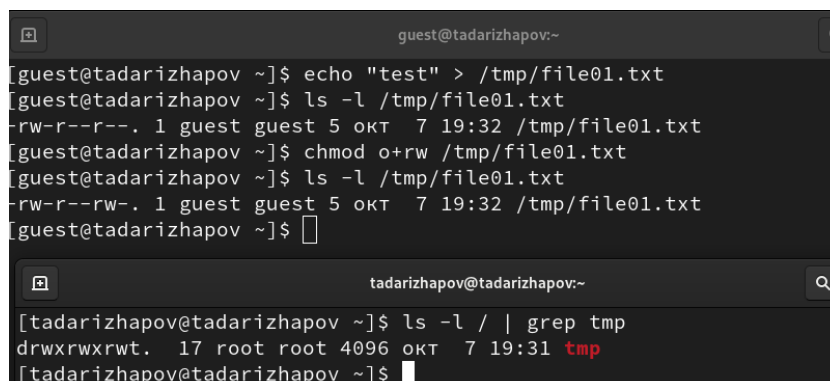
    while (bytes_read == sizeof (buffer));
    close (fd);
    return 0;
}

guest@tadarizhapov lab05$ ./readfile /etc/shadow
root:$6$uW4z.u2pzQZ915C3Lr1vqFNig6Cs2QfRMC2k4XwEEZLtz4089HFcq5uDgkeARCMGDNPS5pwN1BxrtUjYyDGZyr6mbRE7sX1KZ.:10:99999:7:::
bin:*:19469:0:99999:7:::
daemon:*:19469:0:99999:7:::
adm:*:19469:0:99999:7:::
lp:*:19469:0:99999:7:::
sync:*:19469:0:99999:7:::
shutdown:*:19469:0:99999:7:::
```

Figure 3.13: Запуск программы readfile

2 часть: Исследование Sticky-бита

1)Командой “ls -l / | grep tmp” убеждаемся, что атрибут Sticky на директории /tmp установлен. От имени пользователя guest создаём файл file01.txt в директории /tmp со словом test командой “echo test” > /tmp/file01.txt”. Просматриваем атрибуты у только что созданного файла и разрешаем чтение и запись для категории пользователей “все остальные” командами “ls -l /tmp/file01.txt” и “chmod o+rw /tmp/file01.txt” (Рисунок 3.14).



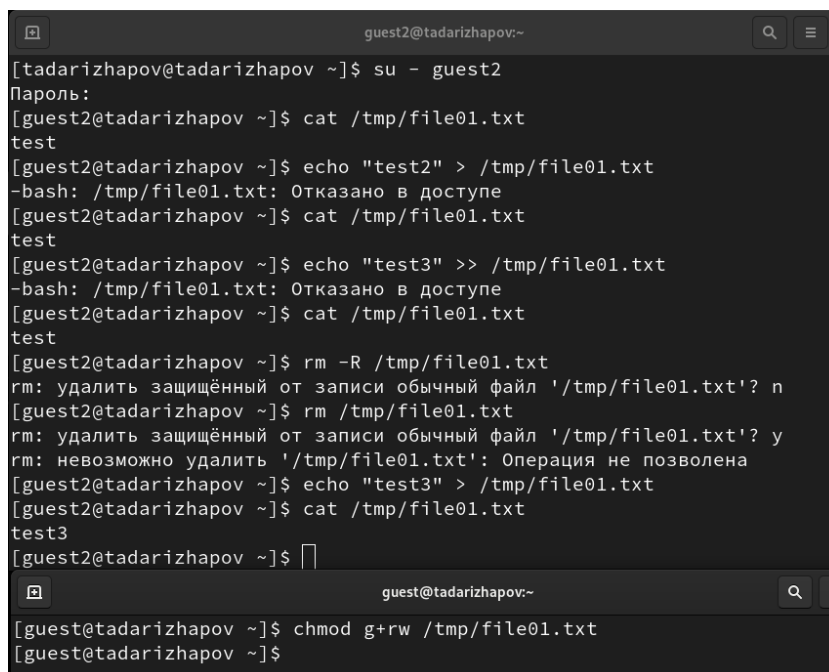
```
guest@tadarizhapov:~$ echo "test" > /tmp/file01.txt
guest@tadarizhapov:~$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 окт 7 19:32 /tmp/file01.txt
guest@tadarizhapov:~$ chmod o+rw /tmp/file01.txt
guest@tadarizhapov:~$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 окт 7 19:32 /tmp/file01.txt
guest@tadarizhapov:~$

tadarizhapov@tadarizhapov:~$ ls -l / | grep tmp
drwxrwxrwt. 17 root root 4096 окт 7 19:31 tmp
tadarizhapov@tadarizhapov:~$
```

Figure 3.14: Создание файла file01.txt

2)От имени пользователя guest2 пробуем прочитать файл командой “cat /tmp/file01.txt” - это удалось. Далее пытаемся дозаписать в файл слово test2, проверить содержимое файла и записать в файл слово test3, стереv при этом

всю имеющуюся в файле информацию - эти операции удалось выполнить только в случае, если еще дополнительно разрешить чтение и запись для группы пользователей командой “chmod g+rw /tmp/file01.txt”. От имени пользователя guest2 пробуем удалить файл - это не удастся ни в каком из случаев, возникает ошибка (Рисунок 3.15).



```
guest2@tadarizhapov:~  
[tadarizhapov@tadarizhapov ~]$ su - guest2  
Пароль:  
[guest2@tadarizhapov ~]$ cat /tmp/file01.txt  
test  
[guest2@tadarizhapov ~]$ echo "test2" > /tmp/file01.txt  
-bash: /tmp/file01.txt: Отказано в доступе  
[guest2@tadarizhapov ~]$ cat /tmp/file01.txt  
test  
[guest2@tadarizhapov ~]$ echo "test3" >> /tmp/file01.txt  
-bash: /tmp/file01.txt: Отказано в доступе  
[guest2@tadarizhapov ~]$ cat /tmp/file01.txt  
test  
[guest2@tadarizhapov ~]$ rm -R /tmp/file01.txt  
rm: удалить защищенный от записи обычный файл '/tmp/file01.txt'? n  
[guest2@tadarizhapov ~]$ rm /tmp/file01.txt  
rm: удалить защищенный от записи обычный файл '/tmp/file01.txt'? y  
rm: невозможно удалить '/tmp/file01.txt': Операция не позволена  
[guest2@tadarizhapov ~]$ echo "test3" > /tmp/file01.txt  
[guest2@tadarizhapov ~]$ cat /tmp/file01.txt  
test3  
[guest2@tadarizhapov ~]$  
guest@tadarizhapov:~  
[guest@tadarizhapov ~]$ chmod g+rw /tmp/file01.txt  
[guest@tadarizhapov ~]$
```

Figure 3.15: Попытка выполнить действия над файлом file01.txt от имени пользователя guest2

3)Повышаем права до суперпользователя командой “su -” и выполняем команду, снимающую атрибут t с директории /tmp “chmod -t /tmp”. После чего покидаем режим суперпользователя командой “exit”. Повторяем предыдущие шаги. Теперь нам удаётся удалить файл file01.txt от имени пользователя, не являющегося его владельцем (Рисунок 3.16).

```

[guest2@tadarizhapov ~]$ su -
Пароль:
[root@tadarizhapov ~]# chmod -t /tmp
[root@tadarizhapov ~]# exit
выход
[guest2@tadarizhapov ~]$ ls -l / | grep tmp
drwxrwxrwx. 17 root root 4096 окт 7 19:40 tmp
[guest2@tadarizhapov ~]$ cat /tmp/file01.txt
test3
[guest2@tadarizhapov ~]$ echo "test2" >> /tmp/file01.txt
[guest2@tadarizhapov ~]$ cat /tmp/file01.txt
test3
test2
[guest2@tadarizhapov ~]$ echo "test3" > /tmp/file01.txt
[guest2@tadarizhapov ~]$ cat /tmp/file01.txt
test3
[guest2@tadarizhapov ~]$ rm /tmp/file01.txt
[guest2@tadarizhapov ~]$ ls -l /tmp
итого 0
srwxrwxrwx. 1 gdm gdm 0 окт 3 22:53 dbus-N08DK7CLCV
srwxrwxrwx. 1 gdm gdm 0 сен 30 19:05 dbus-yuHlBbivKT

```

Figure 3.16: Удаление атрибута t (Sticky-бита) и повторение действий

4)Повышаем свои права до суперпользователя и возвращаем атрибут t на директорию /tmp (Рисунок 3.17).

```

[guest2@tadarizhapov ~]$ su -
Пароль:
[root@tadarizhapov ~]# chmod +t /tmp
[root@tadarizhapov ~]# exit
выход
[guest2@tadarizhapov ~]$ ls -l / | /tmp
-bash: /tmp: это каталог
[guest2@tadarizhapov ~]$ ls -l / | grep tmp
drwxrwxrwt. 17 root root 4096 окт 7 19:43 tmp
[guest2@tadarizhapov ~]$ █

```

Figure 3.17: Возвращение атрибута t (Sticky-бита)

4 Выводы

- В ходе выполнения данной лабораторной работы я изучил механизмы изменения идентификаторов, применение SetUID- и Sticky-битов. Получил практические навыки работы в консоли с дополнительными атрибутами. Рассмотрел работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

5 Список литературы

- Стандартные права SetUID, SetGID, Sticky в Linux [Электронный ресурс]. URL: <https://linux-notes.org/standartny-e-prava-unix-suid-sgid-sticky-bity/>.