

# **Лабораторная работа №5**

**Основы Информационной Безопасности**

Чистов Даниил Максимович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Подготовка к лабораторной работы . . . . .	6
3.2	Исследование SetUID и SetGID битов . . . . .	6
3.3	Исследование Sticky-битов . . . . .	12
<b>4</b>	<b>Выводы</b>	<b>15</b>
<b>5</b>	<b>Список литературы</b>	<b>16</b>

# Список иллюстраций

3.1	Подготовка к выполнению . . . . .	6
3.2	Создание файла simpleid.c . . . . .	7
3.3	Код и вывод программы simpleid . . . . .	7
3.4	Работа программы simpleid2 . . . . .	7
3.5	Команда смены владельца и SetUID бит . . . . .	8
3.6	Новый владелец и атрибуты программы simpleid2 . . . . .	8
3.7	simpleid2 и id . . . . .	8
3.8	simpleid2 и id при SetGID-бите . . . . .	9
3.9	readfile.c . . . . .	9
3.10	Смена прав и владельца readfile . . . . .	10
3.11	Попытка прочесть readfile.c программой от лица суперпользователя . . . . .	11
3.12	Попытка прочесть readfile.c программой от лица guest . . . . .	11
3.13	Попытка прочесть /etc/shadow программой . . . . .	12
3.14	Наличие атрибута sticky у директории tmp . . . . .	12
3.15	Успешно добавленные атрибуты у file01.txt . . . . .	12
3.16	Проверка работы атрибутов файла file01.txt . . . . .	13
3.17	guest2 не может удалить file01.txt . . . . .	13
3.18	Снятие атрибута t с директории tmp . . . . .	13
3.19	Новые возможности пользователя guest2 . . . . .	14
3.20	Возвращение атрибута t . . . . .	14

# 1 Цель работы

Целью данной лабораторной работы является 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

1. Исследование SetUID и SetGID битов
2. Исследование Sticky-битов

## 3 Выполнение лабораторной работы

### 3.1 Подготовка к лабораторной работы

Перед выполнением требуется выяснить, установлен ли компилятор gcc. Всё установлено. Также нужно отключить систему запретов командой `setenforce 0`, после чего вывод команды `getenforce` становится “Permissive” (рис. 3.1).

```
[dmchistov@dmchistov ~]$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/8/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-redhat-linux
Configured with: ../configure --enable-bootstrap --enable-languages=c,c++,fortran,lto --prefix=/usr --mandir
=/usr/share/man --infodir=/usr/share/info --with-bugurl=https://bugs.rockylinux.org/ --enable-shared --enabl
e-threads=posix --enable-checking=release --enable-multilib --with-system-zlib --enable-_cxa_atexit --disab
le-libunwind-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-gcc-major-version-only --
with-linker-hash-style=gnu --enable-plugin --enable-initfini-array --with-lsl --disable-libmpx --enable-offl
oad-targets=nvptx-none --without-cuda-driver --enable-gnu-indirect-function --enable-cet --with-tune=generic
--with-arch_32=x86-64 --build=x86_64-redhat-linux
Thread model: posix
gcc version 8.5.0 20210514 (Red Hat 8.5.0-22) (GCC)
[dmchistov@dmchistov ~]$ setenforce 0
setenforce: setenforce() failed
[dmchistov@dmchistov ~]$ sudo setenforce 0
[sudo] password for dmchistov:
[dmchistov@dmchistov ~]$ getenforce
Permissive
```

Рис. 3.1: Подготовка к выполнению

### 3.2 Исследование SetUID и SetGID битов

Вхожу в систему как пользователь `guest` и создаю файл `simpleid.c` (рис. 3.2).

```
[dmchistov@dmchistov ~]$ su guest
Password:
[guest@dmchistov dmchistov]$ touch simpleid.c
touch: cannot touch 'simpleid.c': Permission denied
[guest@dmchistov dmchistov]$ cd ~
[guest@dmchistov ~]$ touch simpleid.c
[guest@dmchistov ~]$
```

Рис. 3.2: Создание файла simpleid.c

Вставляю в файл код из задания, компилирую, запускаю, после чего прописываю команду `id` и сравниваю результаты. Всё сходится (рис. 3.3).



```
simpleid.c
admin:///home/guest

#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@dmchistov ~]$ gcc simpleid.c -o simpleid
[guest@dmchistov ~]$ ./simpleid
uid=1001, gid=1001
[guest@dmchistov ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@dmchistov ~]$
```

Рис. 3.3: Код и вывод программы simpleid

Усложняю программу, вставив новый код из задания, сохраняю файл как `simpleid2.c`, компилирую и запускаю (рис. 3.4).



```
simpleid2.c
admin:///home/guest

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

Рис. 3.4: Работа программы simpleid2

Прописываю в консоль следующие команды, как на скриншоте ниже (рис. 3.5).

Первая команда меняет владельца файла на root. Теперь суперпользователь является владельцем файла. Вторая команда устанавливает SetUID-бит, т.е. программа будет запускаться от имени владельца файла, а не от того, кто её запустил.

```
guest@dmchistov:~  
dmchistov@dmchistov ~]$ sudo chown root:guest /home/guest/simpleid2  
sudo] password for dmchistov:  
dmchistov@dmchistov ~]$ sudo chmod u+s /home/guest/simpleid2  
dmchistov@dmchistov ~]$
```

Рис. 3.5: Команда смены владельца и SetUID бит

В предисловии к данной лабораторной работе сказано: “любая bash-программа интерпретируется в процессе своего выполнения, т.е. существует сторонняя программа-интерпретатор, которая выполняет считывание файла сценария и выполняет его последовательно. Сам интерпретатор выполняется с правами пользователя, его запустившего, а значит, и выполняемая программа использует эти права.”

Проверяю правильно ли установились новые атрибуты и сменился ли владелец у программы simpleid2 - всё верно (рис. 3.6).

```
[guest@dmchistov ~]$ ls -l simpleid2  
-rwsrwxr-x. 1 root guest 18312 Apr 18 23:15 simpleid2  
[guest@dmchistov ~]$
```

Рис. 3.6: Новый владелец и атрибуты программы simpleid2

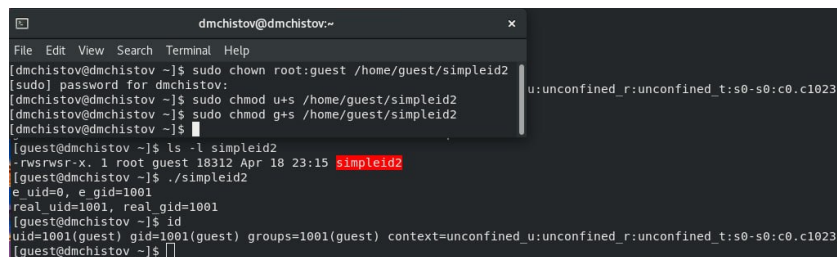
Теперь запускаю simpleid2 и id, сравниваю результат. e\_uid стал равен 0 при выполнении программы simpleid2 (рис. 3.7).

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

Рис. 3.7: simpleid2 и id

Тоже самое проделываю относительно SetGID-бит, т.е. в этот раз пишу не chmod u+s, а chmod g+s (рис. 3.8).

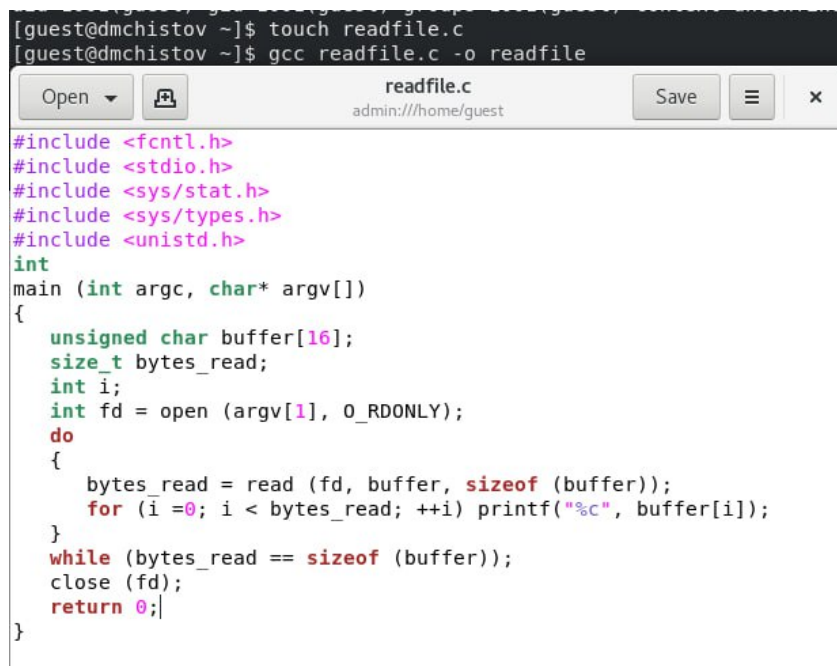




```
dmchistov@dmchistov:~  
File Edit View Search Terminal Help  
[dmchistov@dmchistov ~]$ sudo chown root:guest /home/guest/simpleid2  
[sudo] password for dmchistov:  
[dmchistov@dmchistov ~]$ sudo chmod u+s /home/guest/simpleid2  
[dmchistov@dmchistov ~]$ sudo chmod g+s /home/guest/simpleid2  
[dmchistov@dmchistov ~]$  
[guest@dmchistov ~]$ ls -l simpleid2  
-rwsrwsr-x. 1 root guest 18312 Apr 18 23:15 simpleid2  
[guest@dmchistov ~]$ ./simpleid2  
e uid=0, e gid=1001  
[guest@dmchistov ~]$ id  
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
[guest@dmchistov ~]$
```

Рис. 3.8: simpleid2 и id при SetGID-бите

Создаю новую программу readfile.c, беру код из задания, компилирую программу (рис. 3.9).



```
[guest@dmchistov ~]$ touch readfile.c  
[guest@dmchistov ~]$ gcc readfile.c -o readfile  
  
readfile.c  
admin:///home/guest  
Save  
x  
#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. 3.9: readfile.c

Делаю так, чтобы только суперпользователь мог читать файл readfile.c, а guest не мог. Также изменяю владельца этого файла. На скриншоте ниже видно, как в нижней консоли, пользователь guest не может прочитать этот файл (рис. 3.10).



```
[root@dmchistov guest]# chown root /home/guest/readfile
[root@dmchistov guest]# chmod u+S /home/guest/readfile
chmod: invalid mode: 'u+S'
Try 'chmod --help' for more information.
[root@dmchistov guest]# chmod u+s /home/guest/readfile
[root@dmchistov guest]# cd /home/guest/
[root@dmchistov guest]# ./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;
}
[root@dmchistov guest]#
```

Рис. 3.11: Попытка прочесть readfile.c программой от лица суперпользователя

Однако, если запустить её в консоли пользователя guest, то нам выдают сообщение об ошибке - недостаточно прав (рис. 3.12).

```
[guest@dmchistov ~]$ ./readfile readfile.c
bash: ./readfile: Permission denied
[guest@dmchistov ~]$
```

Рис. 3.12: Попытка прочесть readfile.c программой от лица guest

Аналогичная ситуация происходит при попытке прочесть /etc/shadow (рис. 3.13).

```
[root@dmchistov guest]# ./readfile /etc/shadow
root:$6$LC63CJU.pGjTWZI0$aG\qtqPPxIJLVZ2cuFY9Ru0ss5rz89lIVg8EVQoa1hLQEp9gGwEXRIwxdyc3..5NgKMLUsgFFF1sKy
bin:*.19767:0:99999:7:::
daemon:*.19767:0:99999:7:::
adm:*.19767:0:99999:7:::
lp:*.19767:0:99999:7:::
sync:*.19767:0:99999:7:::
shutdown:*.19767:0:99999:7:::
halt:*.19767:0:99999:7:::
mail:*.19767:0:99999:7:::
operator:*.19767:0:99999:7:::

[guest@dmchistov ~]$ ./readfile /etc/shadow
bash: ./readfile: Permission denied
[guest@dmchistov ~]$
```

Рис. 3.13: Попытка прочесть /etc/shadow программой

### 3.3 Исследование Sticky-битов

Выясняю, установлен ли sticky-бит на директорию /tmp. Да, установлен, т.к. в конце сть буква t (рис. 3.14).

```
[guest@dmchistov ~]$ ls -l / | grep tmp
drwxrwxrwt. 8 root root 4096 Apr 19 00:02 tmp
[guest@dmchistov ~]$
```

Рис. 3.14: Наличие атрибута sticky у директории tmp

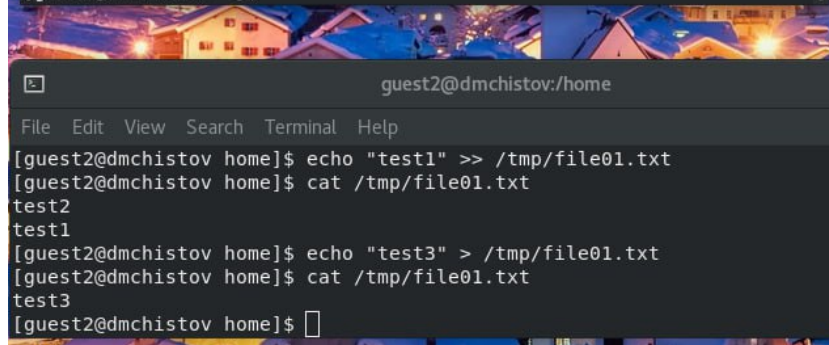
От имени пользователя guest создаю файл file01.txt со словом “test”, затем разрешаю чтение и запись для категории пользователей “все остальные” для данного файла (рис. 3.15).

```
[guest@dmchistov ~]$ echo "test" > /tmp/file01.txt
[guest@dmchistov ~]$ ls -l /tmp/file01.txt
-rw-rw-r--. 1 guest guest 5 Apr 19 00:09 /tmp/file01.txt
[guest@dmchistov ~]$ chmod o+rw /tmp/file01.txt
[guest@dmchistov ~]$ ls -l /tmp/file01.txt
-rw-rw-rw-. 1 guest guest 5 Apr 19 00:09 /tmp/file01.txt
[guest@dmchistov ~]$
```

Рис. 3.15: Успешно добавленные атрибуты у file01.txt

От пользователя guest2 пытаюсь прочитать файл, дозаписать в него слово и переписать его полностью. Всё работает, следовательно мы успешно установили атрибуты в прошлом шаге (рис. 3.16).

```
[guest@dmchistov ~]$ ls -l /tmp/file01.txt
-rw-rw-rw-. 1 guest guest 5 Apr 19 00:09 /tmp/file01.txt
[guest@dmchistov ~]$
```

A terminal window titled 'guest2@dmchistov:/home' with a menu bar (File, Edit, View, Search, Terminal, Help). The terminal shows the following commands and output:

```
[guest2@dmchistov home]$ echo "test1" >> /tmp/file01.txt
[guest2@dmchistov home]$ cat /tmp/file01.txt
test2
test1
[guest2@dmchistov home]$ echo "test3" > /tmp/file01.txt
[guest2@dmchistov home]$ cat /tmp/file01.txt
test3
[guest2@dmchistov home]$
```

Рис. 3.16: Проверка работы атрибутов файла file01.txt

Однако, удалить этот файл от лица guest2 не получается (рис. 3.17).

```
[guest2@dmchistov home]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest2@dmchistov home]$
```

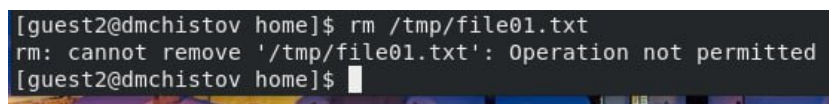
A terminal window titled 'guest2@dmchistov:/home' showing the command 'rm /tmp/file01.txt' and the error message 'rm: cannot remove '/tmp/file01.txt': Operation not permitted'.

Рис. 3.17: guest2 не может удалить file01.txt

От лица суперпользователя снимаю атрибут t (Sticky-бит) с директории /tmp, после чего командой “ls -l / | grep tmp” проверяю, что атрибута t действительно больше нет (рис. 3.18).

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

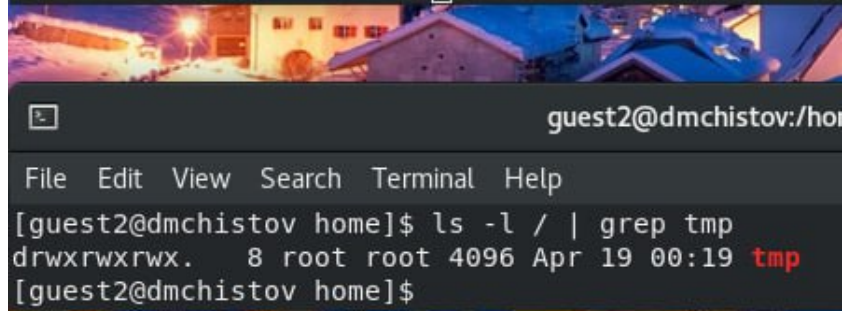
A terminal window titled 'guest2@dmchistov:/home' showing the command 'ls -l / | grep tmp' and the output 'drwxrwxrwx. 8 root root 4096 Apr 19 00:19 tmp'.

Рис. 3.18: Снятие атрибута t с директории tmp

Пытаюсь повторить все предыдущие шаги. Guest2 до сих пор может прочитать

файл, дозаписать в него что-либо, переписать его, но теперь он также может его удалить, благодаря тому, что мы сняли атрибут `t` с папки, в которой лежит этот файл (рис. 3.19).

```
[guest2@dmchistov home]$ ls -l / | grep tmp
drwxrwxrwx. 8 root root 4096 Apr 19 00:19 tmp
[guest2@dmchistov home]$ echo "test1" >> /tmp/file01.txt
[guest2@dmchistov home]$ cat /tmp/file01.txt
test3
test1
[guest2@dmchistov home]$ echo "test3" > /tmp/file01.txt
[guest2@dmchistov home]$ cat /tmp/file01.txt
test3
[guest2@dmchistov home]$ rm /tmp/file01.txt
[guest2@dmchistov home]$ ls /tmp/file01.txt
ls: cannot access '/tmp/file01.txt': No such file or directory
[guest2@dmchistov home]$ ls /tmp
systemd-private-4ad1197b0ada443ba1f905670361f9c0-colord.service-USQeP3
systemd-private-4ad1197b0ada443ba1f905670361f9c0-ModemManager.service-vil47h
systemd-private-4ad1197b0ada443ba1f905670361f9c0-rtkit-daemon.service-3qIFw3
```

Рис. 3.19: Новые возможности пользователя `guest2`

Папка `tmp` довольно важна, поэтому лучше атрибут `t` вернуть на место, после наших экспериментов (рис. 3.20).

```
[root@dmchistov ~]# chmod +t /tmp
[root@dmchistov ~]# exit
logout
[dmchistov@dmchistov ~]$
```

Рис. 3.20: Возвращение атрибута `t`

## 4 Выводы

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

## **5 Список литературы**

Лабораторная работа №5