

Лабораторная работа №5

Основы информационной безопасности

Чистов Д. М.

19 апреля 2025

Российский университет дружбы народов, Москва, Россия

Объединённый институт ядерных исследований, Дубна, Россия

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

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

Выполнение лабораторной работы

Подготовка к лабораторной работы

Перед выполнение требуется выяснить, установлен ли компилятор gcc. Всё установлено.

Также нужно отключить систему запретов командой `setenforce 0`, после чего вывод команды `getenforce` становится “Permissive”.

```
[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-isl --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. 1: Подготовка к выполнению

Вхожу в систему как пользователь guest и создаю файл simpleid.c.

```
[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. 2: Создание файла simpleid.c

Исследование SetUID и SetGID битов

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



```
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;
}

C Tab Width: 8 Ln 13, Col 1 INS

[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: Код и вывод программы simpleid

Исследование SetUID и SetGID битов

Усложняю программу, вставив новый код из задания, сохраняю файл как simpleid2.c, компилирую и запускаю.



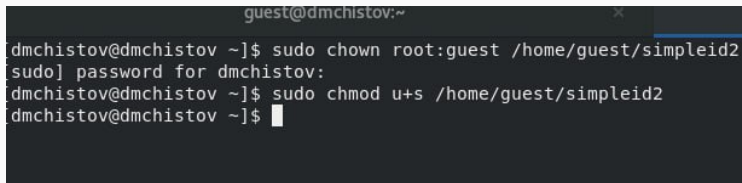
```
Open [icon] simpleid2.c Save [icon]  
admin:///home/guest  
#include <sys/types.h>  
#include <unistd.h>  
#include <stdio.h>  
int  
main ()  
{  
    uid_t real_uid = getuid ();  
    uid_t e_uid = geteuid ();  
    gid_t real_gid = getgid ();  
    gid_t e_gid = getegid ();  
    printf ("e_uid=%d, e_gid=%d\n", e_uid, e_gid);  
    printf ("real_uid=%d, real_gid=%d\n", real_uid, real_gid);  
    return 0;  
}
```

```
C Tab Width: 8 Ln 10, Col 30  
[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. 4: Работа программы simpleid2

Прописываю в консоль следующие команды, как на скриншоте ниже.

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.

A screenshot of a terminal window with a dark background. The window title is 'guest@dmchistov:~'. The terminal shows the following sequence of commands and output:

```
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. 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.

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

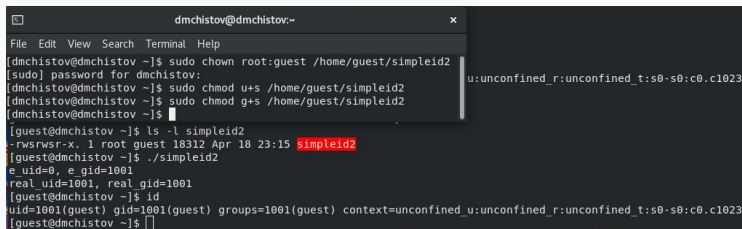
Рис. 6: Новый владелец и атрибуты программы simpleid2

Теперь запускаю simpleid2 и id, сравниваю результат. e_uid стал равен 0 при выполнении программы simpleid2.

```
[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. 7: simpleid2 и id

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



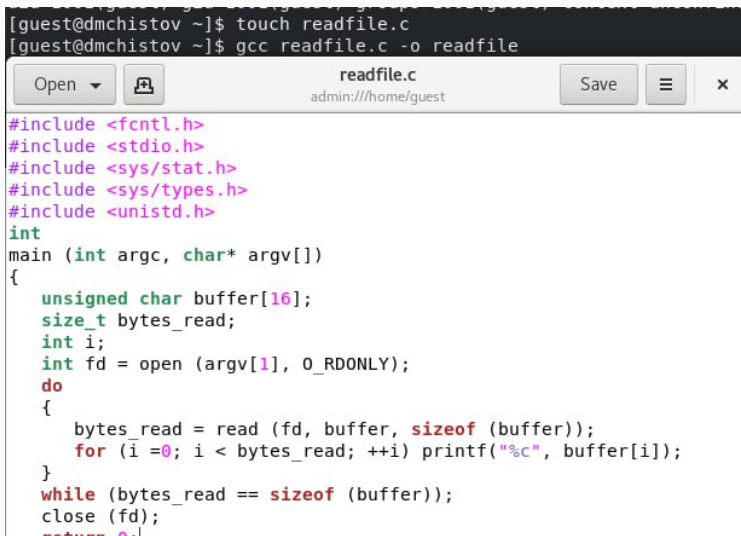
```
dmchistov@dmchistov:~  
File Edit View Search Terminal Help  
[dmchistov@dmchistov ~]$ sudo chown root:guest /home/guest/simpleid2  
[sudo] password for dmchistov:  
[dmchistov@dmchistov ~]$ sudo chmod u+s /home/guest/simpleid2  
[dmchistov@dmchistov ~]$ sudo chmod g+s /home/guest/simpleid2  
[dmchistov@dmchistov ~]$  
[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  
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. 8: simpleid2 и id при SetGID-бите

Исследование SetUID и SetGID битов

Создаю новую программу readfile.c, беру код из задания, компилирую программу.

```
[guest@dmchistov ~]$ touch readfile.c
[guest@dmchistov ~]$ gcc readfile.c -o readfile
```



```
#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;
}
```

Исследование SetUID и SetGID битов

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.

```
root@dmchistov:/home/guest

File Edit View Search Terminal Help

[dmchistov@dmchistov ~]$ sudo chown root:guest /home/guest/readfile
[sudo] password for dmchistov:
[dmchistov@dmchistov ~]$ chmod 700 /home/guest/readfile
chmod: cannot access '/home/guest/readfile': Permission denied
[dmchistov@dmchistov ~]$ sudo chmod 700 /home/guest/readfile
[dmchistov@dmchistov ~]$ sudo su
[root@dmchistov dmchistov]# /home/guest
bash: /home/guest: Is a directory
[root@dmchistov dmchistov]# cd /home/guest
[root@dmchistov dmchistov]# cat readfile
ELF...@...lib64/ld-linux-x86-64.so.2...libc.so
..._6putcharreadcloseopen _libc start_mainGLIBC 2.2.5 ITM deregisterTMCloneTable_gmon_start__ITM_reg
isterTMCloneTable...
H00t...
0%0

guest@dmchistov:~

File Edit View Search Terminal Help

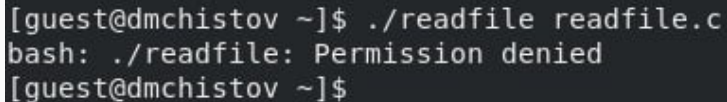
[guest@dmchistov ~]$ ls -l readfile
-rwx-----. 1 root guest 18256 Apr 18 23:37 readfile
```


Исследование SetUID и SetGID битов

Теперь меняю владельца у скомпилированной программы readfile и устанавливаю SetUID-бит. Проверяю, может ли наша программа прочитать файл readfile.c. Если запускать её в консоли суперпользователя, то она, естественно всё прочитает.

```
[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));
```

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

A terminal window with a dark background and light gray text. The prompt is [guest@dmchistov ~]\$. The user enters ./readfile readfile.c. The response is bash: ./readfile: Permission denied. The prompt returns to [guest@dmchistov ~]\$.

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

Рис. 12: Попытка прочесть readfile.c программой от лица guest

Аналогичная ситуация происходит при попытке прочесть `/etc/shadow`.



```
[root@dmchistov guest]# ./readfile /etc/shadow
root:$6$LC63CJU.pGjTWZI0$aGlqtqPPxIJLVZ2cuFY9Ru0ss5rz89iIVg8EVQoaihLQEp9gGwEXRIwxdyc3..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. 13: Попытка прочесть `/etc/shadow` программой

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

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

Рис. 14: Наличие атрибута sticky у директории tmp

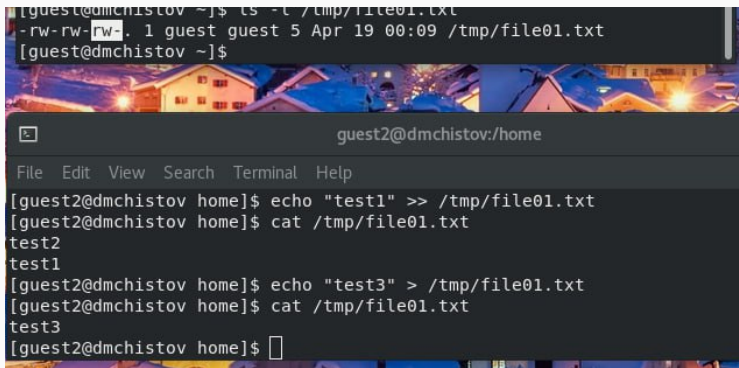
От имени пользователя guest создаю файл file01.txt со словом “test”, затем разрешаю чтение и запись для категории пользователей “все остальные” для данного файла.

```
[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. 15: Успешно добавленные атрибуты у file01.txt

Исследование Sticky-битов

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



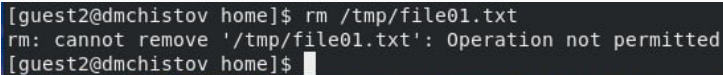
```
[guest@dmchistov ~]$ ls -l /tmp/file01.txt
-rw-rw-rw-. 1 guest guest 5 Apr 19 00:09 /tmp/file01.txt
[guest@dmchistov ~]$
```



```
guest2@dmchistov:/home
File Edit View Search Terminal Help
[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. 16: Проверка работы атрибутов файла file01.txt

Однако, удалить этот файл от лица guest2 не получается.

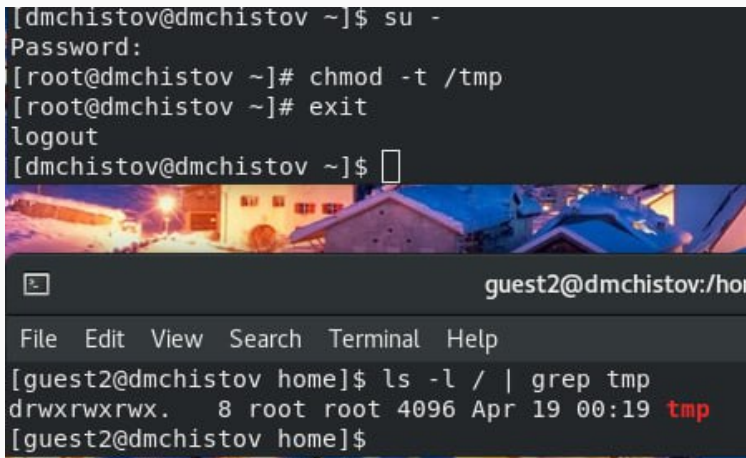
A terminal window with a dark background and light-colored text. The prompt is [guest2@dmchistov home]\$. The command rm /tmp/file01.txt is entered. The output is rm: cannot remove '/tmp/file01.txt': Operation not permitted. The prompt [guest2@dmchistov home]\$ is shown again with a cursor at the end.

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

Рис. 17: guest2 не может удалить file01.txt

Исследование Sticky-битов

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.



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

The screenshot shows a terminal window with a dark background and a colorful, abstract image at the bottom. The terminal text shows a user switching to root, removing the sticky bit from /tmp, and then exiting. Below this, a new terminal window is shown with the user 'guest2' at the 'home' directory. The terminal text shows the user running the command 'ls -l / | grep tmp', which returns the output 'drwxrwxrwx. 8 root root 4096 Apr 19 00:19 tmp'. The word 'tmp' is highlighted in red in the original image.

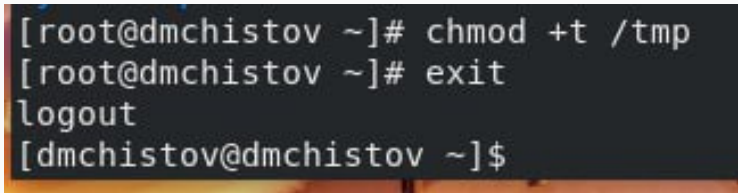
```
guest2@dmchistov:/home  
File Edit View Search Terminal Help  
[guest2@dmchistov home]$ ls -l / | grep tmp  
drwxrwxrwx. 8 root root 4096 Apr 19 00:19 tmp  
[guest2@dmchistov home]$
```


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.

```
[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-US0eP3
systemd-private-4ad1197b0ada443ba1f905670361f9c0-ModemManager.service-vIl47h
systemd-private-4ad1197b0ada443ba1f905670361f9c0-rtkit-daemon.service-3qIFw3
```

Рис. 19: Новые возможности пользователя guest2

Папка tmp довольно важна, поэтому лучше атрибут t вернуть на место, после наших экспериментов.

A terminal window with a dark background and light-colored text. The text shows a root user at a machine named dmchistov. The user runs 'chmod +t /tmp' to restore the sticky bit. Then they type 'exit' and 'logout'. The prompt changes from '[root@dmchistov ~]#' to '[dmchistov@dmchistov ~]\$' after logging out.

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

Рис. 20: Возвращение атрибута t

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

Список литературы

Лабораторная работа №5