

# Отчет по лабораторной работе №5

Основы информационной безопасности

Ничипорова Елена Дмитриевна

## Содержание

Цель работы .....	1
Теоретическое введение .....	1
Выполнение лабораторной работы .....	2
Выводы.....	8
Список литературы.....	8

## 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.

## Теоретическое введение

### 1. Дополнительные атрибуты файлов Linux

В Linux существует три основных вида прав — право на чтение (read), запись (write) и выполнение (execute), а также три категории пользователей, к которым они могут применяться — владелец файла (user), группа владельца (group) и все остальные (others). Но, кроме прав чтения, выполнения и записи, есть еще три дополнительных атрибута. [a]

### Sticky bit

Используется в основном для каталогов, чтобы защитить в них файлы. В такой каталог может писать любой пользователь. Но, из такой директории пользователь может удалить только те файлы, владельцем которых он является. Примером может служить директория /tmp, в которой запись открыта для всех пользователей, но нежелательно удаление чужих файлов.

### SUID (Set User ID)

Атрибут исполняемого файла, позволяющий запустить его с правами владельца. В Linux приложение запускается с правами пользователя, запустившего указанное приложение. Это обеспечивает дополнительную безопасность т.к. процесс с

правами пользователя не сможет получить доступ к важным системным файлам, которые принадлежат пользователю root.

### **SGID (Set Group ID)**

Аналогичен suid, но относится к группе. Если установить sgid для каталога, то все файлы созданные в нем, при запуске будут принимать идентификатор группы каталога, а не группы владельца, который создал файл в этом каталоге.

### **Обозначение атрибутов sticky, suid, sgid**

Специальные права используются довольно редко, поэтому при выводе программы ls -l символ, обозначающий указанные атрибуты, закрывает символ стандартных прав доступа.

Пример: rwsrwsrwt

где первая s — это suid, вторая s — это sgid, а последняя t — это sticky bit

В приведенном примере не понятно, rwt — это rw- или rwx? Определить это просто. Если t маленькое, значит x установлен. Если T большое, значит x не установлен. То же самое правило распространяется и на s.

В числовом эквиваленте данные атрибуты определяются первым символом при четырехзначном обозначении (который часто опускается при назначении прав), например в правах 1777 — символ 1 обозначает sticky bit. Остальные атрибуты имеют следующие числовое соответствие:

- 1 – установлен sticky bit
- 2 – установлен sgid
- 4 – установлен suid

## **2. Компилятор GCC**

GCC - это свободно доступный оптимизирующий компилятор для языков C, C++. Собственно программа gcc это некоторая надстройка над группой компиляторов, которая способна анализировать имена файлов, передаваемые ей в качестве аргументов, и определять, какие действия необходимо выполнить. Файлы с расширением .cc или .C рассматриваются, как файлы на языке C++, файлы с расширением .c как программы на языке C, а файлы с расширением .o считаются объектными [@gcc].

## **Выполнение лабораторной работы**

Для лабораторной работы необходимо проверить, установлен ли компилятор gcc, команда gcc -v позволяет это сделать. Также осуществляется отключение системы запретом с помощью setenforce 0 (рис. 1).

```

Complete!
[root@localhost guest]# 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 guest]# whereis g++
g++:
[root@localhost guest]# gcc -v
Using built-in specs.
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
Target: x86_64-redhat-linux
Configured with: ../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 --enable-gnu-unique-object --enable-linker-build-id --with-gcc-major-version-only --enable-plugin --enable-initfini-array --with-lto-is1 --enable-multilib --with-linker-hash-style=gnu --enable-offload-targets=nvptx-none --without-cuda-driver --enable-gnu-ld-indirect-function --enable-cet --with-tune=generic --with-arch_64=x86-64-v2 --with-arch_32=x86-64 --build=x86_64-redhat-linux --with-build-config=bootstrap-lto --enable-link-serialization=1
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 11.4.1 20231218 (Red Hat 11.4.1-3) (GCC)
[root@localhost guest]#

```

## Подготовка к лабораторной работе

Осуществляется вход от имени пользователя guest (рис. 2).

```

[root@localhost guest]# su guest
[guest@localhost ~]# touch simplified.c
[guest@localhost ~]# nano simplified.c

```

## Вход от имени пользователя guest

Создание файла simplified.c и запись в файл кода (рис. 2)

```

[root@localhost guest]# su guest
[guest@localhost ~]# touch simplified.c
[guest@localhost ~]# nano simplified.c

```

## Создание файла

C++ Листинг 1 #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; }

Содержимое файла выглядит следующи образом (рис. 3)

```

GNU nano 5.6.1 simplified.c
# include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int
main()
{
uid_t uid = geteuid();
gid_t git=getegit();
printf("uid=%d,git=%d\n", uid, git);
return 0;
}

```

## Содержимое файла

Компилирую файл, проверяю, что он скомпилировался (рис. 4)

```
[guest@localhost ~]$ gcc simplified.c -o simplified
[guest@localhost ~]$
```

### Компиляция файла

Запускаю исполняемый файл. В выводе файла выписаны номера пользователя и групп, от вывода при вводе if, они отличаются только тем, что информации меньше

Создание, запись в файл и компиляция файла simplified2.c. Запуск программы (рис. 5)

```
[guest@localhost ~]$ ls
dir dir1 simplified simplified.c test
[guest@localhost ~]$ ./simplified
uid=1000,gid=1000
[guest@localhost ~]$ id
uid=1000(guest) gid=1000(guest) groups=1000(guest) context=unconfined_u:unconfined_r:unconfined_t:s0:c0,c1023
[guest@localhost ~]$
```

### Создание и компиляция файла

C++ Листинг 2 #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. 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;
}
```

### Содержимое файла

С помощью chown изменяю владельца файла на суперпользователя, с помощью chmod изменяю права доступа (рис. 7)

```

[sudo] password for guest:
guest is not in the sudoers file. This incident will be reported.
[guest@localhost ~]$ su root
Password:
[root@localhost guest]# sudo chown root: gurst /home/guest/simpl
chown: cannot access 'gurst': No such file or directory
[root@localhost guest]# sudo chown root: guest /home/guest/simpl
chown: cannot access 'guest': No such file or directory
[root@localhost guest]# sudo chmod u+s: guest /home/guest/simpl
chmod: invalid mode: 'u+s:'
Try 'chmod --help' for more information.

```

### Смена владельца файла и прав доступа к файлу

Сравнение вывода программы и команды id, наша команда снова вывела только ограниченное количество информации (рис. 8)

```

[root@localhost guest]# sudo /home/guest/simplified2
e_uid=0, e_gid = 0
rel_uid=0, real_gid=0
[root@localhost guest]# sudo id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-
[root@localhost guest]#

```

### Запуск файла

#### Создание и компиляция файла readfile.c

C++ Листинг 3

```

#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. 9)

```

GNU nano 5.6.1 touchfile.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) print ("%c", buffer[i]);
    }
    while (bytes_read == sizeof(buffer));
    close (fd);
    return 0;
}

```

### Содержимое файла

```
[root@localhost guest]# sudo chown root:guest /home/root/readfile.c
chown: cannot access '/home/root/readfile.c': No such file or directory
[root@localhost guest]# sudo chmod 700 /home/root/readfile.c
chmod: cannot access '/home/root/readfile.c': No such file or directory
```

Проверка прочесть файл от имени пользователя guest.Прочесть файл не удастся (рис. 11)

```
cat: readfile.c: Отказано в доступе
```

Попытка прочесть тот же файл с помощью программы readfile, в ответ получаем “отказано в доступе” (рис. 12)

```
pe 00000000 00000000 00000000 00000000
d 00000000 d 00000000 d 00000000 00000000 00000000 00000000 00000000 e 00000000 e 00000000 e 00000000 00000000
00000000 1 00000000 1 00000000 00000000 00000000 00000000 00000000 m 00000000 m 00000000 m 00000000 o 00000000
00000000 @ 08 pe
```

Попытка прочесть файл `\etc\shadow` с помощью программы, все еще получаем отказ в доступе (рис. 13)

s-  
 t-  
 |  
 ^

Пробуем прочесть эти же файлы от имени суперпользователя и чтение файлов проходит успешно (рис. 14)

```
root:$6$3reywnb0G.0EfHL7$1td/ZD0qRQQEdbaZehnNr0Kq7LhY9HS4Ip0CdU6H/hmKbVhFsqs02
gd3/YkGPnmw5AD2t0THLZyUxi4eD/rU0::0:99999:7::
bin:*:19469:0:99999:7::
daemon:*:19469:0:99999:7::
adm:*:19469:0:99999:7::
lp:*:19469:0:99999:7::
```

Проверяем папку tmp на наличие атрибута Sticky, т.к. в выводе есть буква t, то атрибут установлен (рис. 15)

```
[root@localhost guest]# ls -l / | grep tmp
```

От имени пользователя guest создаю файл с текстом, добавляю права на чтение и запись для других пользователей (рис. 16)



```
[root@localhost guest]# echo "test" > /tmp/file01.txt
[root@localhost guest]# ls -l /tmp/file01.txt
-rw-r--r--. 1 root root 5 Sep 16 16:43 /tmp/file01.txt
[root@localhost guest]#
```

{#fig:016width=70%}

Вхожу в систему от имени пользователя guest2, от его имени могу прочитать файл file01.txt, но перезаписать информацию в нем не могу (рис. 17)

```
[root@localhost guest]# su guest2
[guest2@localhost guest]$ cat /tmp/file01.txt
test
[guest2@localhost guest]$ echo "test2">> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@localhost guest]$ cat /tmp/file01.txt
test
[guest2@localhost guest]$ _
```

### *Попытка чтения файла*

Также невозможно добавить в файл file01.txt новую информацию от имени пользователя guest2 (рис. 18)

```
[guest2@localhost guest]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@localhost guest]$ cat /tmp/file01.txt
test
[guest2@localhost guest]$
```

### *Попытка записи в файл*

Далее пробуем удалить файл, снова получаем отказ (рис. 19)

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

### *Попытка удалить файл*

От имени суперпользователя снимаем с директории атрибут Sticky (рис. 20)

```
[root@localhost ~]# su guest2
[guest2@localhost root]$ ls -l / | grep tmp
drwxrwxrwx. 5 root root 4096 Sep 16 16:48 tmp
[guest2@localhost root]$
```

### *Смена атрибутов файла*

Проверяем, что атрибут действительно снят (рис. 21)

```
[root@localhost ~]# su guest2
[guest2@localhost root]$ ls -l / | grep tmp
drwxrwxrwx. 5 root root 4096 Sep 16 16:48 tmp
[guest2@localhost root]$ ls -l /cat /tmp/file01.txt
ls: cannot access '/cat': No such file or directory
-rw-r--r--. 1 root root 5 Sep 16 16:43 /tmp/file01.txt
[guest2@localhost root]$
```

### *Проверка атрибутов директории*

Далее был выполнен повтор предыдущих действий. По результатам без Sticky-бита запись в файл и дозапись в файл осталась невозможной, зато удаление файла прошло успешно (рис. 22)

```
[lguest2@localhost root]$ su -  
Password:  
[root@localhost ~]# chmod +t /tmp  
[root@localhost ~]#
```

*Повтор предыдущих действий*

Возвращение директории tmp атрибута t от имени суперпользователя (рис. 22)

```
[lguest2@localhost root]$ su -  
Password:  
[root@localhost ~]# chmod +t /tmp  
[root@localhost ~]#
```

*Изменение атрибутов*

## Выводы

Изучила механизм изменения идентификаторов, применила SetUID- и Sticky-биты. 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