

Лабораторная работа №5

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Кармацкий Н. С. Группа НФИбд-01-21

29 Сентября 2024

Российский университет дружбы народов, Москва, Россия

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

SetUID (SUID) SetUID — это бит, который позволяет пользователям запускать исполняемые файлы с правами владельца этого файла. Это означает, что если файл имеет установленный бит SUID, любой пользователь, запускающий этот файл, будет выполнять его с привилегиями владельца файла, а не с привилегиями своего собственного пользователя. Это часто используется для программ, требующих повышенных прав, например, команда passwd, которая позволяет пользователям изменять свои пароли.

Sticky-бит Sticky-бит — это бит, который устанавливается на каталоги и ограничивает возможность удаления файлов в этом каталоге. Если Sticky-бит установлен на каталог, только владелец файла или суперпользователь может удалить или переименовать файлы в этом каталоге. Это полезно для общих каталогов, таких как /tmp, где множество пользователей могут создавать файлы.

Выполнение лабораторной работы

Проверка компилятора

1. Проверяем наличие компилятора языка С (рис. 1)

```
[guest@nskarmatskiy dir1]$ whereis gcc
gcc: /usr/bin/gcc /usr/lib/gcc /usr/libexec/gcc /usr/share/man/man1/gcc.1.gz /us
r/share/info/gcc.info.gz
[guest@nskarmatskiy dir1]$ whereis g++
g++: /usr/bin/g++ /usr/share/man/man1/g++.1.gz
[guest@nskarmatskiy dir1]$ gcc -v
Используются внутренние спецификации.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/aarch64-redhat-linux/11/lto-wrapper
Целевая архитектура: aarch64-redhat-linux
Параметры конфигурации: ../configure --enable-bootstrap --enable-host-pie --enab
le-host-bind-now --enable-languages=c,c++,fortran,lto --prefix=/usr --mandir=/us
r/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-ob
ject --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-gnu-indirect-function --build=aarch64-redhat-linux --with-build-config
=bootstrap-lto --enable-link-serialization=l
Модель многопоточности: posix
Supported LTO compression algorithms: zlib zstd
gcc версия 11.4.1 20231218 (Red Hat 11.4.1-3) (GCC)
[guest@nskarmatskiy dir1]$
```

Рис. 1: Наличие компилятора

Создание программы 1

2. Создаем отдельный каталог для программ, а так же файл с 1 первой программой (рис. 2)

```
[guest@nskarmatskiy ~]$ mkdir lab5
[guest@nskarmatskiy ~]$ cd lab5/
[guest@nskarmatskiy lab5]$ touch simpleid.c
[guest@nskarmatskiy lab5]$ █
```

Рис. 2: Создание файлов

Создание программы 2

3. Заполнение файла (рис. 3)

```
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 }
```

Рис. 3: Листинг программы

Создание программы 3

4. Компилируем и выполняем программу, а так же выполним системую программму id. В результате получем из нашей программы те же данные, что и из id, но только в более коротком варианте (рис. 4)

```
[guest@nskarmatskiy lab5]$ gcc simpleid.c -o simpleid
[guest@nskarmatskiy lab5]$ ls
simpleid simpleid.c
[guest@nskarmatskiy lab5]$ ./simpleid
uid=1001, gid=1001
[guest@nskarmatskiy lab5]$ id
uid=1001(guest) gid=1001(guest) группа=1001(guest) контекст=unconfined_u:unconfi
ned_r:unconfined_t:s0-s0:c0.c1023
[guest@nskarmatskiy lab5]$
```

Рис. 4: Запуск программы

Создание программы 4

5. Усложняем программу, добавив вывод действительных идентификаторов, так же назовем ее simpleid2 (рис. 5)



The screenshot shows a code editor window with the file name "simpleid2.c" at the top. The code itself is as follows:

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

Рис. 5: Усложненная программа

6. Компилируем новую программу и получаем дополнительные сведения в отличии от первоначальной программы (рис. 6)

```
[guest@nskarmatskiy lab5]$ gcc simpleid2.c -o simpleid2
[guest@nskarmatskiy lab5]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
```

Рис. 6: Компиляция нового файла

Изменение прав доступа 1

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

```
[root@nskarmatskiy lab5]# chown root:guest /home/guest/lab5/simpleid2
[root@nskarmatskiy lab5]# chmod u+s /home/guest/lab5/simpleid2
[root@nskarmatskiy lab5]# ls -l simpleid2
-rwsr-xr-x. 1 root guest 79112 сен 1 16:17 simpleid2
```

Рис. 7: Новые права

Далее запускаем программу получаем все значения равными 0, так как выполняем все от прав суперпользователя

Изменение прав доступа 2

- Пробуем сменить атрибуты относительно SetGID-бита, получаем все точно тоже самое (рис. 8)

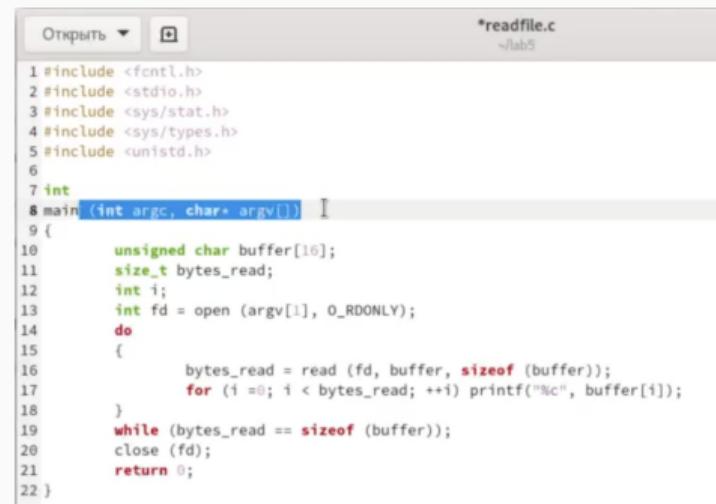
```
[root@nskarmatskiy lab5]# chown root:guest /home/guest/lab5/simpleid2
[root@nskarmatskiy lab5]# chmod g+s /home/guest/lab5/simpleid2
[root@nskarmatskiy lab5]# ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=0
[root@nskarmatskiy lab5]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 8: Другие права доступа

Создание новой программы 1

9. Создадим программу `readfile.c`, которая будет читать файлы.

Откомпилируем ее, а так же поменяем владельца у файла и изменим права так, чтобы только суперпользователь мог прочитать его, а guest не мог (рис. 10)



```
Открыть ▾ *readfile.c
~/lab5

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     while (bytes_read == sizeof (buffer));
20     close (fd);
21     return 0;
22 }
```

Рис. 9: Листинг программы

Создание новой программы 2

```
[guest@nskarmatskiy lab5]$ touch readfile.c
[guest@nskarmatskiy lab5]$ gcc readfile.c -o readfile
[guest@nskarmatskiy lab5]$ su nskarmatskiy
[nskarmatskiy@nskarmatskiy lab5]$ sudo -i
bash: sudo: команда не найдена...
[nskarmatskiy@nskarmatskiy lab5]$ sudo -i
[root@nskarmatskiy ~]# ls
anaconda-ks.cfg
[root@nskarmatskiy ~]# cd /home/guest/lab5/
[root@nskarmatskiy lab5]# chown root:guest readfile
[root@nskarmatskiy lab5]# chmod 700 readfile
[root@nskarmatskiy lab5]# chmod -r readfile.c
[root@nskarmatskiy lab5]# chmod u+s readfile
```

Рис. 10: Компиляция и изменение в правах файла

Проверка работоспособности программы с новыми правами 1

10. Проверка новых прав доступа для guest, как видим ничего не получается сделать (рис. 11)

```
[guest@nskarmatskiy lab5]$ cat readfile  
cat: readfile: Отказано в доступе  
[guest@nskarmatskiy lab5]$ cat readfile.c  
cat: readfile.c: Отказано в доступе
```

Рис. 11: Отказано в доступе

Проверка работоспособности программы с новыми правами 2

пытаемся считать файл с помощью программы, но получаю отказ в доступе (рис. 12)

```
[guest@nskarmatskiy lab5]$ ./readfile readfile.c
bash: ./readfile: Отказано в доступе
[guest@nskarmatskiy lab5]$ ./readfile /etc/shadow
bash: ./readfile: Отказано в доступе
```

Рис. 12: Отказано в доступе при работе программы

Мы запретили всем, кроме суперпользователя использовать или читать файл, поэтому никто и не может выполнить никакое действие с файлом

Исследование Sticky-бита 1

11. Выясним, установлен ли атрибут Sticky на директории /tmp. От имени пользователя guest создаем файл file01.txt с тестом text в каталоге /tmp. Так же просмотрим атрибуты файла и разрешим чтение и запись для категории пользователей “все остальные” (рис. 13)

```
[guest@nskarmatskiy lab5]$ ls -l / | grep tmp
drwxrwxrwt. 18 root root 4096 сен 1 16:30 tmp
[guest@nskarmatskiy lab5]$ echo "test" > /tmp/file01.txt
[guest@nskarmatskiy lab5]$ cat /tmp/file01.txt
test
[guest@nskarmatskiy lab5]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 сен 1 16:32 /tmp/file01.txt
[guest@nskarmatskiy lab5]$ chmod o+rw /tmp/file01.txt
[guest@nskarmatskiy lab5]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 сен 1 16:32 /tmp/file01.txt
```

Рис. 13: Создание и настройка файла

Исследование Sticky-бита 2

12. Попробуем прочитать, записать и удалить файл от имени пользователя guest2. Получаем отказ в доступе везде, кроме чтения файла (рис. 14)

```
[guest@nskarmatskiy lab5]$ su guest2
Пароль:
[guest2@nskarmatskiy lab5]$ cat /tmp/file01.txt
test
[guest2@nskarmatskiy lab5]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@nskarmatskiy lab5]$ cat /tmp/file01.txt
test
[guest2@nskarmatskiy lab5]$ rm /tmp/file01.txt
rm: удалить защищённый от записи обычный файл '/tmp/file01.txt'? у
rm: невозможно удалить '/tmp/file01.txt': Операция не позволена
```

Рис. 14: Попытка чтения и записи в файл от guest2

Исследование Sticky-бита 3

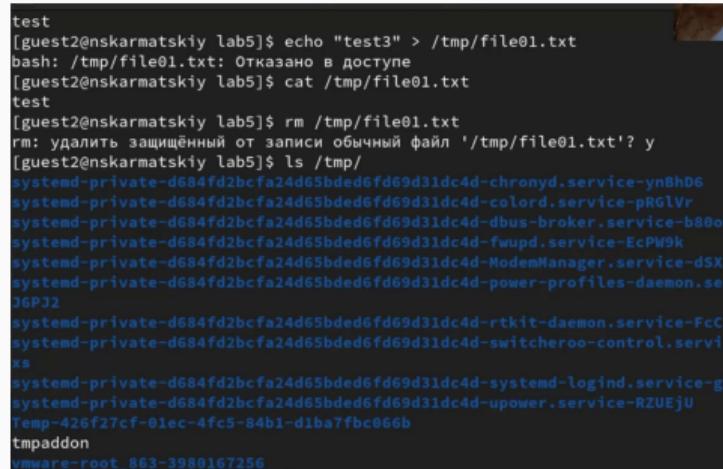
13. Снимем атрибут Sticky с директории /tmp и повторим действия, описанные в пункте 12. (рис. 15)

```
[guest2@nskarmatskiy lab5]$ su nskarmatskiy
[nskarmatskiy@nskarmatskiy lab5]$ sudo -i
[root@nskarmatskiy ~]# chmod -t /tmp
[root@nskarmatskiy ~]# exit
выход
```

Рис. 15: Снятие атрибута Sticky с директории /tmp

Исследование Sticky-бита 4

Попробуем снова выполнить все те же действия. Получаем доступ к файлу от guest2 (рис. 16)



```
test
[guest2@nskarmatskiy lab5]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@nskarmatskiy lab5]$ cat /tmp/file01.txt
test
[guest2@nskarmatskiy lab5]$ rm /tmp/file01.txt
rm: удалить защищенный от записи обычный файл '/tmp/file01.txt'? у
[guest2@nskarmatskiy lab5]$ ls /tmp/
systemd-private-d684fd2bcfa24d65bded6fd69d31dc4d-chronyd.service-yn8hD6
systemd-private-d684fd2bcfa24d65bded6fd69d31dc4d-colord.service-prGlVr
systemd-private-d684fd2bcfa24d65bded6fd69d31dc4d-dbus-broker.service-b88o8
systemd-private-d684fd2bcfa24d65bded6fd69d31dc4d-fwupd.service-EcPM9k
systemd-private-d684fd2bcfa24d65bded6fd69d31dc4d-ModemManager.service-dSX8
systemd-private-d684fd2bcfa24d65bded6fd69d31dc4d-power-profiles-daemon.ser
J6P32
systemd-private-d684fd2bcfa24d65bded6fd69d31dc4d-rtkit-daemon.service-FcCj
systemd-private-d684fd2bcfa24d65bded6fd69d31dc4d-switcheroo-control.servic
s
systemd-private-d684fd2bcfa24d65bded6fd69d31dc4d-systemd-logind.service-g1
systemd-private-d684fd2bcfa24d65bded6fd69d31dc4d-upower.service-RZUEjU
Temp-426f27cf-01ec-4fc5-84b1-d1ba7fbcb066b
tmpaddon
mware-root 863-3980167256
```

Рис. 16: Попытка чтения и записи в файл от guest2

14. Вернем первоначальные атрибуты директории /tmp(рис. 17)

```
[nskarmatskiy@nskarmatskiy lab5]$ sudo -i  
[root@nskarmatskiy ~]# chm  
chmem chmod  
[root@nskarmatskiy ~]# chmod +t /tmp  
[root@nskarmatskiy ~]# exit  
выход
```

Рис. 17: Восстановление атрибутов

Выводы

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