

Отчет по лабораторной работе №5

Основы информационной безопасности

Мажитов Магомед Асхабович, НКНбд-01-21

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	8
4	Выводы	16
5	Список литературы. Библиография	17

Список иллюстраций

3.1	Компилятор	8
3.2	Создание simpleid.c	8
3.3	Компиляция simpleid.c	9
3.4	Создание simpleid2.c	9
3.5	Сравнение	10
3.6	Манипуляции simpleid2	10
3.7	Сравнение	11
3.8	Изменение владельца readfile	12
3.9	Попытка прочитать readfile	12
3.10	Попытка запустить readfile	12
3.11	Проверка наличия Sticky атрибута	13
3.12	Создание file01.txt	13
3.13	Манипуляции с file01.txt	13
3.14	Попытка удалить file01.txt	14
3.15	Снятие атрибута t	14
3.16	Манипуляции с file01.txt	14
3.17	Попытка удалить file01.txt	15

Список таблиц

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

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. [u?]

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?].

3 Выполнение лабораторной работы

1 Проверил установлен ли компилятор gcc и g++.

```
[mamazhitov@localhost ~]$ gcc -v
Используются внутренние спецификации.
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
Целевая архитектура: x86_64-redhat-linux
Параметры конфигурации: ../configure --enable-bootstrap --enable-languages=c,c++
,fortran,lto --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --w
ith-bugurl=https://bugs.rockylinux.org/ --enable-shared --enable-threads=posix -
-enable-checking=release --enable-multilib --with-system-zlib --enable-_cxa_ate
xit --disable-libunwind-exceptions --enable-gnu-unique-object --enable-linker-bu
ild-id --with-gcc-major-version-only --with-linker-hash-style=gnu --enable-plugi
n --enable-initfini-array --with-isl --disable-libmpx --enable-offload-targets=n
vptx-none --without-cuda-driver --enable-gnu-indirect-function --enable-cet --wi
th-tune=generic --with-arch_32=x86-64 --build=x86_64-redhat-linux
Модель многопоточности: posix
gcc версия 8.5.0 20210514 (Red Hat 8.5.0-22) (GCC)
[mamazhitov@localhost ~]$ whereis g++
g++: /usr/bin/g++ /usr/share/man/man1/g++.1.gz
[mamazhitov@localhost ~]$
```

Рис. 3.1: Компилятор

2 Вошел в систему от имени пользователя guest и создал программу simpleid.c.

```
[guest@localhost home]$ cd guest
[guest@localhost ~]$ touch simpleid.c
[guest@localhost ~]$ nano simpleid.c
```

Рис. 3.2: Создание simpleid.c

```
#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;
}

```

3 Скомпилировал программу и убедился, что файл программы создан. Далее запустил исполнительный файл, а также ввел системную программу *id* для дальнейшего сравнения выводов.

```

[guest@localhost ~]$ gcc simpleid.c -o simpleid
[guest@localhost ~]$ ls
dir1 simpleid simpleid.c
[guest@localhost ~]$ nano simpleid.c
[guest@localhost ~]$ ./simpleid
uid=1001, gid=1001
[guest@localhost ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfi
ned_r:unconfined_t:s0-s0:c0.c1023
[guest@localhost ~]$

```

Рис. 3.3: Компиляция simpleid.c

Результаты идентичны.

4 Создал программу *simpleid2.c*.

```

[guest@localhost ~]$ touch simpleid2.c
[guest@localhost ~]$ nano simpleid2.c
[guest@localhost ~]$

```

Рис. 3.4: Создание simpleid2.c

```

#include <sys/types.h>
#include <unistd.h>

```

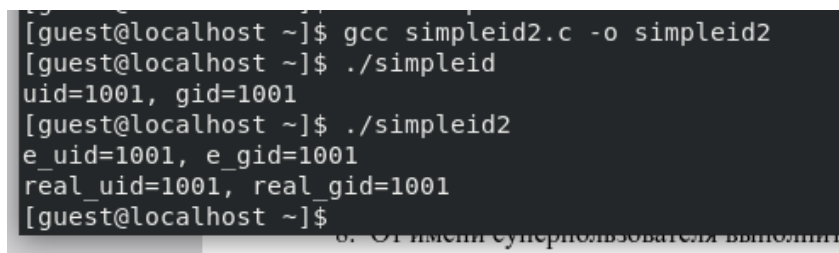
```

#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;
}

```

5 Скомпилировал программу и сравнил выводы прошлой и новой программ.



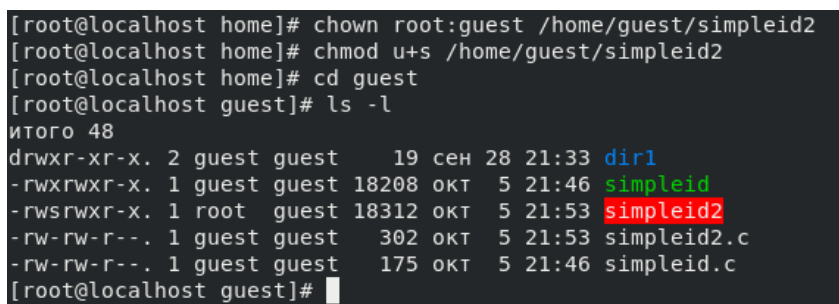
```

[guest@localhost ~]$ gcc simpleid2.c -o simpleid2
[guest@localhost ~]$ ./simpleid
uid=1001, gid=1001
[guest@localhost ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@localhost ~]$

```

Рис. 3.5: Сравнение

6 Далее я поменял владельца файла *simpleid2* и изменил права доступа к нему.



```

[root@localhost home]# chown root:guest /home/guest/simpleid2
[root@localhost home]# chmod u+s /home/guest/simpleid2
[root@localhost home]# cd guest
[root@localhost guest]# ls -l
итого 48
drwxr-xr-x. 2 guest guest   19 сен 28 21:33 dirl
-rwxrwxr-x. 1 guest guest 18208 окт  5 21:46 simpleid
-rwsrwxr-x. 1 root  guest 18312 окт  5 21:53 simpleid2
-rw-rw-r--. 1 guest guest   302 окт  5 21:53 simpleid2.c
-rw-rw-r--. 1 guest guest   175 окт  5 21:46 simpleid.c
[root@localhost guest]#

```

Рис. 3.6: Манипуляции simpleid2

7 Запустил *simpleid2* и *id*.

```
[guest@localhost ~]$ ./simpleid2
e uid=0, e gid=1001
real uid=1001, real gid=1001
[guest@localhost ~]$ id
uid=1001(guest) gid=1001(guest) rpnны=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@localhost ~]$
```

Рис. 3.7: Сравнение

Как мы видим после изменения владельца *simpleid2*, вывод программы изменился.

8 Создал программу *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;
}
```

Скомпилировал файл и далее также изменил владельца *readfile* и права доступа к нему, так, чтобы только суперпользователь(root) мог прочитать его, а guest не мог.

```
[root@localhost guest]# chown root:guest /home/guest/readfile.c
[root@localhost guest]# chmod 700 readfile.c
[root@localhost guest]# chmod -u readfile.c
[root@localhost guest]# chmod u+s readfile.c
[root@localhost guest]# ls -l
итого 72
drwxr-xr-x. 2 guest guest   19 сен 28 21:33 dirl
-rwxrwxr-x. 1 guest guest 18256 окт  5 22:02 readfile
---S----- 1 root  guest   402 окт  5 22:01 readfile.c
-rwxrwxr-x. 1 guest guest 18208 окт  5 21:46 simpleid
-rwsrwxr-x. 1 root  guest 18312 окт  5 21:53 simpleid2
-rw-rw-r-- 1 guest guest   302 окт  5 21:53 simpleid2.c
-rw-rw-r-- 1 guest guest   175 окт  5 21:46 simpleid.c
[root@localhost guest]#
```

Рис. 3.8: Изменение владельца readfile

9 Попробовал прочитать файл от имени *guest*.

```
[guest@localhost ~]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@localhost ~]$
```

Рис. 3.9: Попытка прочитать readfile

Попытка не увенчалась успехом.

10 Проверил, может ли программа *readfile* прочитать файл *readfile.c* и */etc/shadow*.

```
[root@localhost guest]# ./readfile readfile.c
[root@localhost guest]# eof (buffer);("%c", buffer[i]);
[root@localhost guest]# ./readfile /etc/shadow
root:$6$UpDUSH.jEJNgJAn8$IHiyNMum3xzH4a7yb67E0uYWBzBt9WB.9ZyQKYyJA1V30.p9ojwqAai
20ck80vhxGmGN7vXeMBuCPPncY6ojv.:0:99999:7:::
bin:*.19767:0:99999:7:::
daemon:*.19767:0:99999:7:::
adm:*.19767:0:99999:7:::
```

Рис. 3.10: Попытка запустить readfile

11 Проверил, установлен ли атрибут *Sticky* на директории */tmp*.

```
[root@localhost guest]# cd ..  
[root@localhost home]# ls -l |grep tmp  
[root@localhost home]# ls -l / | grep tmp  
drwxrwxrwt. 14 root root 4096 окт  5 22:13 tmp  
[root@localhost home]#
```

Рис. 3.11: Проверка наличия Sticky атрибута

12 От имени пользователя *guest* создал файл *file01.txt* в директории */tmp* со словом *test* и изменил права доступа к нему.

```
[guest@localhost ~]$ echo "test" > /tmp/file01.txt  
[guest@localhost ~]$ ls -l /tmp/file01.txt  
-rw-rw-r--. 1 guest guest 5 окт  5 23:00 /tmp/file01.txt  
[guest@localhost ~]$ chmod o+rw /tmp/file01.txt  
[guest@localhost ~]$ ls -l /tmp/file01.txt  
-rw-rw-rw-. 1 guest guest 5 окт  5 23:00 /tmp/file01.txt  
[guest@localhost ~]$
```

Рис. 3.12: Создание file01.txt

13 От пользователя *guest2* (не являющегося владельцем) попробовал прочитать файл, переписать содержимое файла, а также дописать в файл новые данные.

```
[guest2@localhost mamazhitov]$ cd ..  
[guest2@localhost home]$ cat /tmp/file01.txt  
test  
[guest2@localhost home]$ echo "test2" > /tmp/file01.txt  
[guest2@localhost home]$ cat /tmp/file01.txt  
test2  
[guest2@localhost home]$ S
```

Рис. 3.13: Манипуляции с file01.txt

14 Попробовал удалить файл.

```
test3
[guest2@localhost home]$ rm /tmp/file01.txt
rm: невозможно удалить '/tmp/file01.txt': Операция не позволена
[guest2@localhost home]$
```

Рис. 3.14: Попытка удалить file01.txt

14 Повысив права до суперпользователя, снял атрибут *t*.

```
[root@localhost home]# chmod -t /tmp
[root@localhost home]# ls -l /tmp
итого 4
-rw-rw-rw-. 1 guest      guest      12 окт  5 22:20 file01.txt
drwx----- 3 root       root       17 окт  5 21:34 systemd-private-68
drwx----- 3 root       root       17 окт  5 21:34 systemd-private-68
drwx----- 3 root       root       17 окт  5 21:36 systemd-private-68
drwx----- 3 root       root       17 окт  5 21:34 systemd-private-68
cV
drwx----- 3 root       root       17 окт  5 21:34 systemd-private-68
Z6
drwx----- 2 mamazhitov mamazhitov  6 окт  5 21:37 Temp-561bb308-5a4a
[root@localhost home]# ls -l / | grep tmp
drwxrwxrwx. 14 root root 4096 окт  5 22:21 tmp
[root@localhost home]#
```

Рис. 3.15: Снятие атрибута *t*

14 Повторил действия из пунктов 13-14.

```
[guest2@localhost home]$ ls -l / | grep tmp
drwxrwxrwx. 14 root root 4096 окт  5 22:21 tmp
[guest2@localhost home]$ cat /tmp/file01.txt
test2
test3
[guest2@localhost home]$ echo "test2" > /tmp/file01.txt
[guest2@localhost home]$ cat /tmp/file01.txt
test2
[guest2@localhost home]$ echo "test3" >> /tmp/file01.txt
[guest2@localhost home]$ cat /tmp/file01.txt
test2
test3
[guest2@localhost home]$ rm /tmp/file01.txt
[guest2@localhost home]$ ls /tmp
systemd-private-684e72e4d3364f87b633fd6f3cbd2be3-chrond.service-2xjhx
systemd-private-684e72e4d3364f87b633fd6f3cbd2be3-colord.service-RTxSqC
systemd-private-684e72e4d3364f87b633fd6f3cbd2be3-fwupd.service-vwFWTH
systemd-private-684e72e4d3364f87b633fd6f3cbd2be3-ModemManager.service-BZGxcV
systemd-private-684e72e4d3364f87b633fd6f3cbd2be3-rtkit-daemon.service-h3VPZ6
Temp-561bb308-5a4a-494a-aaaf-5104105da6e7
[guest2@localhost home]$
```

Рис. 3.16: Манипуляции с file01.txt

В этот раз получилось удалить *file01.txt*.

14 Попробовал удалить файл.

```
test3  
[guest2@localhost home]$ rm /tmp/file01.txt  
rm: невозможно удалить '/tmp/file01.txt': Операция не позволена  
[guest2@localhost home]$
```

Рис. 3.17: Попытка удалить file01.txt

4 Выводы

Изучил механизм изменения идентификаторов, применил SetUID- и Sticky-биты. 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. Библиография