

Отчёт по лабораторной работе №5

Основы информационной безопасности

Мурашов Иван Вячеславович

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	8
4	Выводы	17

Список иллюстраций

3.1	Подготовка к лабораторной работе	8
3.2	Вход от имени пользователя guest	8
3.3	Создание файла	8
3.4	Содержимое файла	9
3.5	Компиляция файла	9
3.6	Сравнение команд	9
3.7	Создание и компиляция файла	10
3.8	Содержимое файла	10
3.9	Смена владельца файла и прав доступа к файлу	11
3.10	Запуск файла	11
3.11	Создание и компиляция файла	11
3.12	Содержимое файла	12
3.13	Смена владельца файла и прав доступа к файлу	12
3.14	Попытка прочесть содержимое файла	12
3.15	Попытка прочесть содержимое файла программой	13
3.16	Попытка прочесть содержимое файла программой	13
3.17	Чтение файла от имени суперпользователя	13
3.18	Проверка атрибутов директории tmp	13
3.19	Создание файла, изменение прав доступа	14
3.20	Попытка чтения файла	14
3.21	Попытка записи в файл	14
3.22	Попытка удалить файл	14
3.23	Смена атрибутов файла	15
3.24	Проверка атрибутов директории	15
3.25	Повтор предыдущих действий	16
3.26	Изменение атрибутов	16

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

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

```
ivmurashov@ivmurashov ~]$ 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 --without-isl --enable-multilib --with-linker-hash-style=gnu --enable-offload-targets=nvptx-none --without-cuda-driver --enable-gnu-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 11.5.0 20240719 (Red Hat 11.5.0-5) (GCC)
ivmurashov@ivmurashov ~]$ sudo setenforce
[sudo] password for ivmurashov:
ivmurashov@ivmurashov ~]$ sudo setenforce 0
[sudo] password for ivmurashov:
ivmurashov@ivmurashov ~]$ getenforce
Permissive
```

Рис. 3.1: Подготовка к лабораторной работе

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

```
[ivmurashov@ivmurashov ~]$ sudo su guest
[sudo] password for ivmurashov:
[guest@ivmurashov ivmurashov]$
```

Рис. 3.2: Вход от имени пользователя `guest`

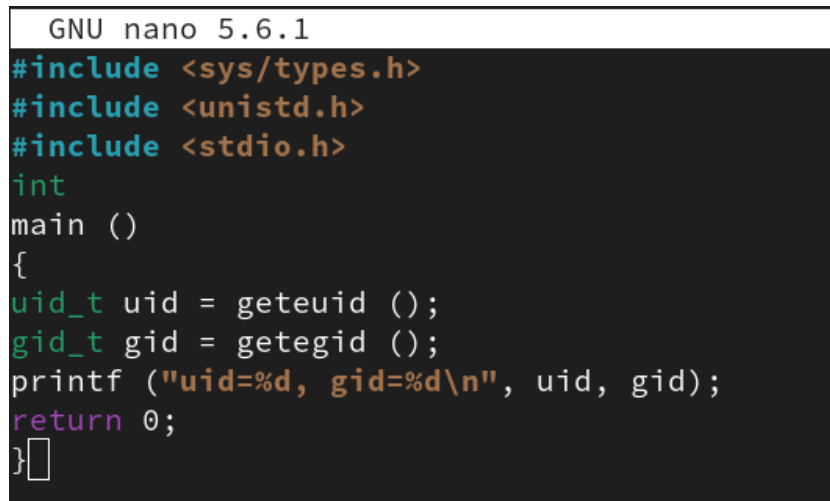
Создание файла `simplified.c` и запись в файл кода (рис. 3)

```
[guest@ivmurashov ~]$ touch simplified.c
[guest@ivmurashov ~]$ nano simplified.c
```

Рис. 3.3: 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 следующти образом (рис. 4)



```
GNU nano 5.6.1
#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;
}
```

Рис. 3.4: Содержимое файла

Компилирую файл, проверяю, что он скомпилировался (рис. 5)



```
[guest@ivmurashov ~]$ gcc simplified.c -o simplified
[guest@ivmurashov ~]$ ls
Desktop  dir1  Documents  Downloads  Music  Pictures  Public  simplified  simplified.c  Templates  test  Videos
```

Рис. 3.5: 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 (рис. 6)



```
[guest@ivmurashov ~]$ ./simplified
uid=1003, gid=1001
[guest@ivmurashov ~]$ id
uid=1003(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 3.6: Сравнение команд

Создание, запись в файл и компиляция файла `simplified2.c`. Запуск программы (рис. 7)

```

[guest@ivmurashov ~]$ touch simplified2.c
[guest@ivmurashov ~]$ nano simplified2.c
[guest@ivmurashov ~]$ gcc simplified2.c -o simplified2
[guest@ivmurashov ~]$ ./simplified2
e_uid=1003, e_gid=1001
real_uid=1003, real_gid=1001

```

Рис. 3.7: 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. 8)

```

GNU nano 5.6.1
#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. 3.8: Содержимое файла

С помощью `chown` изменяю владельца файла на суперпользователя, с помощью `chmod` изменяю права доступа (рис. 9)

```

[ivmurashov@ivmurashov ~]$ sudo chown root:guest /home/guest/simplified2
[sudo] password for ivmurashov:
[ivmurashov@ivmurashov ~]$ sudo chmod u+s /home/guest/simplified2
chmod: cannot access '/home/guest/simplified2': No such file or directory
[ivmurashov@ivmurashov ~]$ sudo chmod u+s /home/guest/simplified2
[ivmurashov@ivmurashov ~]$ sudo ls -l /home/guest/simplified2
-rwsr-xr-x. 1 root guest 17656 Apr 19 02:43 /home/guest/simplified2
[ivmurashov@ivmurashov ~]$

```

Рис. 3.9: Смена владельца файла и прав доступа к файлу

Сравнение вывода программы и команды `id`, наша команда снова вывела только ограниченное количество информации (рис. 10)

```

[ivmurashov@ivmurashov ~]$ sudo /home/guest/simplified2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[ivmurashov@ivmurashov ~]$ id
uid=1000(ivmurashov) gid=1000(ivmurashov) groups=1000(ivmurashov),10(wheel) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[ivmurashov@ivmurashov ~]$ sudo id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023

```

Рис. 3.10: Запуск файла

Создание и компиляция файла `readfile.c` (рис. 11)

```

[guest@ivmurashov ~]$ touch readfile.c
[guest@ivmurashov ~]$ nano readfile.c
[guest@ivmurashov ~]$ gcc readfile.c -o readfile
[guest@ivmurashov ~]$ ls
Downloads Music Pictures Public readfile readfile.c simplified simplified2 simplified2.c simplified.c templates test Videos
[guest@ivmurashov ~]$

```

Рис. 3.11: Создание и компиляция файла

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. 12)

```

GNU nano 5.6.1
#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. 3.12: Содержимое файла

Снова от имени суперпользователя меняю владельца файла readfile. Далее меняю права доступа так, чтобы пользователь guest не смог прочесть содержимое файла (рис. 13)

```

[ivmurashov@ivmurashov ~]$ sudo chown root:guest /home/guest/readfile.c
[ivmurashov@ivmurashov ~]$ sudo chmod u+s /home/guest/readfile.c
[ivmurashov@ivmurashov ~]$ sudo chmod 700 /home/guest/readfile.c
[ivmurashov@ivmurashov ~]$ sudo chmod -r /home/guest/readfile.c
[ivmurashov@ivmurashov ~]$ sudo chmod u+s /home/guest/readfile.c

```

Рис. 3.13: Смена владельца файла и прав доступа к файлу

Проверка прочесть файл от имени пользователя guest. Прочесть файл не удастся (рис. 14)

```

[guest@ivmurashov ~]$ cat readfile.c
cat: readfile.c: Permission denied

```

Рис. 3.14: Попытка прочесть содержимое файла

Попытка прочесть тот же файл с помощью программы readfile, в ответ получаем “отказано в доступе” (рис. 15)


```
[guest@ivmurashov ~]$ echo "test" > /tmp/file01.txt
[guest@ivmurashov ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 Apr 19 02:56 /tmp/file01.txt
[guest@ivmurashov ~]$ chmod o+rw /tmp/file01.txt
[guest@ivmurashov ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 Apr 19 02:56 /tmp/file01.txt
```

Рис. 3.19: Создание файла, изменение прав доступа

Вхожу в систему от имени пользователя guest2, от его имени могу прочитать файл file01.txt, но перезаписать информацию в нем не могу (рис. 20)

```
[ivmurashov@ivmurashov ~]$ su guest2
Password:
[guest2@ivmurashov ivmurashov]$ cd
[guest2@ivmurashov ~]$ cat /tmp/file01.txt
test
[guest2@ivmurashov ~]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@ivmurashov ~]$ cat /tmp/file01.txt
test
[guest2@ivmurashov ~]$
```

Рис. 3.20: Попытка чтения файла

Также невозможно добавить в файл file01.txt новую информацию от имени пользователя guest2 (рис. 21)

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

Рис. 3.21: Попытка записи в файл

Далее пробуем удалить файл, снова получаем отказ (рис. 22)

```
[guest2@ivmurashov ~]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
rm: cannot remove '/tmp/file01.txt': Operation not permitted
```

Рис. 3.22: Попытка удалить файл

От имени суперпользователя снимаем с директории атрибут Sticky (рис. 23)

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

Рис. 3.23: Смена атрибутов файла

Проверяем, что атрибут действительно снят (рис. 24)

```
[guest2@ivmurashov ~]$ ls -l / | grep tmp  
drwxrwxrwx. 36 root root 4096 Apr 19 03:01 tmp
```

Рис. 3.24: 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. 25)

```

[guest2@ivmurashov ~]$ cat /tmp/file01.txt
test
[guest2@ivmurashov ~]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@ivmurashov ~]$ cat /tmp/file01.txt
test
[guest2@ivmurashov ~]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@ivmurashov ~]$ cat /tmp/file01.txt
test
[guest2@ivmurashov ~]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
[guest2@ivmurashov ~]$ ls -l / | grep tmp
drwxrwxrwx. 36 root root 4096 Apr 19 03:02 tmp
[guest2@ivmurashov ~]$ ls -l
total 0
[guest2@ivmurashov ~]$ ls -l /home/guest
total 76
drwxr-xr-x. 2 1001 guest 6 Mar 7 12:53 Desktop
drwxr-xr-x. 2 guest guest 19 Apr 5 14:52 dir1
drwxr-xr-x. 2 1001 guest 6 Mar 7 12:53 Documents
drwxr-xr-x. 2 1001 guest 6 Mar 7 12:53 Downloads
drwxr-xr-x. 2 1001 guest 6 Mar 7 12:53 Music
drwxr-xr-x. 2 1001 guest 6 Mar 7 15:19 Pictures
drwxr-xr-x. 2 1001 guest 6 Mar 7 12:53 Public
-rwxr-xr-x. 1 guest guest 17600 Apr 19 02:50 readfile
--ws----- 1 root guest 402 Apr 19 02:50 readfile.c
-rwxr-xr-x. 1 guest guest 17552 Apr 19 02:40 simplified
-rwsr-xr-x. 1 root guest 17656 Apr 19 02:43 simplified2
-rw-r--r-- 1 guest guest 303 Apr 19 02:42 simplified2.c
-rw-r--r-- 1 guest guest 175 Apr 19 02:40 simplified.c
drwxr-xr-x. 2 1001 guest 6 Mar 7 12:53 Templates
-rw-r--r-- 1 1001 guest 5 Mar 7 13:10 test
drwxr-xr-x. 2 1001 guest 6 Mar 7 12:53 Videos

```

Рис. 3.25: Повтор предыдущих действий

Возвращение директории tmp атрибута t от имени суперпользователя (рис. 26)

```

[guest2@ivmurashov ~]$ su -
Password:
[root@ivmurashov]~# chmod +t /tmp
[root@ivmurashov]~# exit

```

Рис. 3.26: Изменение атрибутов

4 Выводы

Я изучил 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.