

Отчёт по лабораторной работе №5

Дисциплина: Основы информационной безопасности

Мишина Анастасия Алексеевна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Создание программы	6
2.2	Исследование Sticky-бита	10
3	Выводы	13
	Список литературы	14

Список иллюстраций

2.1	Создание файла simpleid.c	6
2.2	Программа в файле simpleid.c	6
2.3	Выполнение программы simpleid и команды id	7
2.4	Добавление вывода действительных идентификаторов	7
2.5	Компиляция и запуск simpleid2	7
2.6	Запуск simpleid2 с SETUID. Сравнение результатов	8
2.7	Запуск simpleid2 с SETGID. Сравнение результатов	8
2.8	Программа readfile	9
2.9	Изменение прав доступа, проверка от имени пользователя guest	9
2.10	Установка SETUID для readfile, проверка	10
2.11	Проверка наличия атрибута Sticky на /tmp, работа с файлом file01.txt	11
2.12	Снятие атрибута t (Sticky-бит), повторение операций	12
2.13	Возвращение атрибута	12

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 [1].

2 Выполнение лабораторной работы

2.1 Создание программы

Входим в систему от имени пользователя `guest` и создаем файл `simpleid.c`, открываем его в редакторе `vi` (рис. 2.1). В файл вписываем код из мануала (рис. 2.2).

A terminal window with a dark background. The prompt is 'guest@aamishina:~'. The user enters 'su guest', then 'cd', then 'touch simpleid.c', and finally 'vi simpleid.c'. The prompt returns to '~\$' after each command.

```
guest@aamishina:~  
[aamishina@aamishina ~]$ su guest  
Password:  
[guest@aamishina aamishina]$ cd  
[guest@aamishina ~]$ touch simpleid.c  
[guest@aamishina ~]$ vi simpleid.c  
[guest@aamishina ~]$
```

Рис. 2.1: Создание файла `simpleid.c`

A terminal window showing the contents of the file 'simpleid.c' in the vim editor. The code includes headers for sys/types.h, unistd.h, and stdio.h. It defines a main function that gets the current user and group IDs and prints them.

```
guest@aamishina:~ — /usr/bin/vim 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;  
}
```

Рис. 2.2: Программа в файле `simpleid.c`

Скомпилируем программу и убедимся, что файл программы создан: `gcc simpleid.c -o simpleid`. Выполняем программу `simpleid`: `./simpleid` и выполняем системную программу `id`: `id`. Видим, что результаты совпадают (рис. 2.3).

```
[guest@aamishina ~]$ gcc simpleid.c -o simpleid
[guest@aamishina ~]$ ls -l
total 28
drwxrwxrwx. 2 guest guest   72 Mar 27 14:11 .
-rwxr-xr-x. 1 guest guest 80672 Apr 11 12:45 simpleid
-rw-r--r--. 1 guest guest  176 Apr 11 12:45 simpleid.c
[guest@aamishina ~]$ ./simpleid
uid=1001, gid=1001
[guest@aamishina ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@aamishina ~]$
```

Рис. 2.3: Выполнение программы simpleid и команды id

Теперь усложним программу, добавив вывод действительных идентификаторов (рис. 2.4).



```

#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. 2.4: Добавление вывода действительных идентификаторов

Компилируем и запускаем simpleid2.c: gcc simpleid2.c -o simpleid2 и ./simpleid2 (рис. 2.5).

```
[guest@aamishina ~]$ vi simpleid.c
[guest@aamishina ~]$ gcc simpleid.c -o simpleid2
[guest@aamishina ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@aamishina ~]$
```

Рис. 2.5: Компиляция и запуск simpleid2

Переходим в режим суперпользователя и выполняем команды: chown root:guest /home/guest/simpleid2 и chmod u+s /home/guest/simpleid2. От имени суперпользователя мы изменили владельца файла и добавили атрибут s, это означает, что пользователь будет выполнять файл с разрешениями владельца

файла. Проверяем правильность установки: `ls -l simpleid2`. Запускаем `simpleid2` и вводим команду `id`: `./simpleid2` и `id`. Теперь владельцем файла является пользователь с `id 0` (root), а изначально владельцем был пользователь с `id 1001` (guest) (рис. 2.6).

```
[guest@aamishina ~]$ su -
Password:
Last login: Wed Mar 27 14:07:45 MSK 2024 on pts/1
Last failed login: Thu Apr 11 12:49:54 MSK 2024 on pts/0
There was 1 failed login attempt since the last successful login.
[root@aamishina ~]#
[root@aamishina ~]#
[root@aamishina ~]# chown root:guest /home/guest/simpleid2
[root@aamishina ~]# chmod u+s /home/guest/simpleid2
[root@aamishina ~]# exit
logout
[guest@aamishina ~]$ ls -l simpleid2
-rwsr-xr-x. 1 root guest 80768 Apr 11 12:49 simpleid2
[guest@aamishina ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@aamishina ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@aamishina ~]$
```

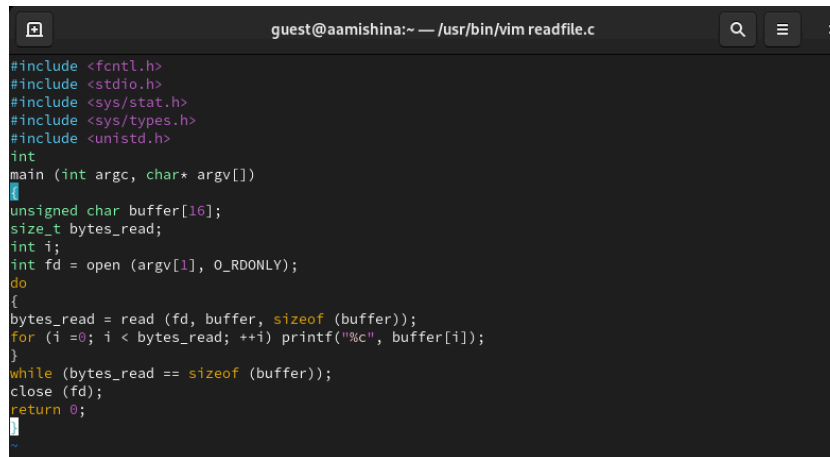
Рис. 2.6: Запуск `simpleid2` с SETUID. Сравнение результатов

Повторяем тоже самое относительно SetGID-бита (рис. 2.7).

```
[guest@aamishina ~]$ su -
Password:
Last login: Thu Apr 11 12:50:01 MSK 2024 on pts/0
[root@aamishina ~]# chmod u-s /home/guest/simpleid2
[root@aamishina ~]# chmod g+s /home/guest/simpleid2
[root@aamishina ~]# exit
logout
[guest@aamishina ~]$ ls -l simpleid2
-rwxr-sr-x. 1 root guest 80768 Apr 11 12:49 simpleid2
[guest@aamishina ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@aamishina ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@aamishina ~]$
```

Рис. 2.7: Запуск `simpleid2` с SETGID. Сравнение результатов

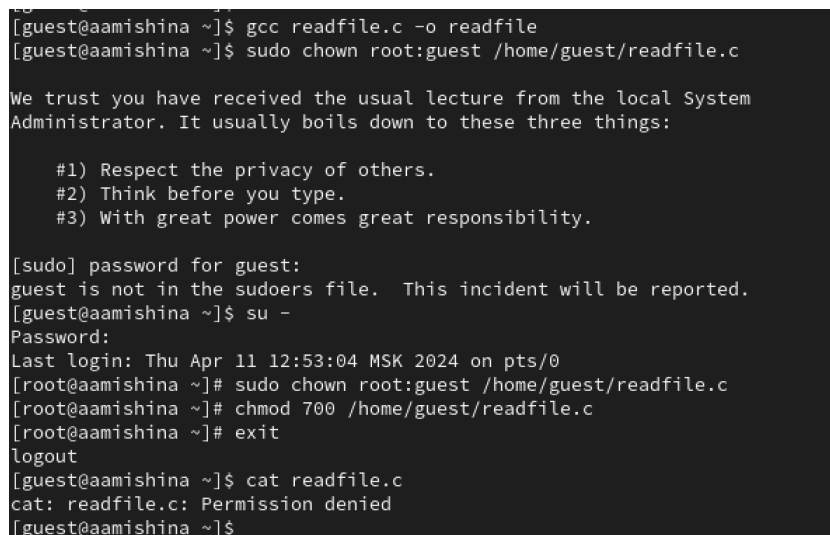
Создаем программу `readfile.c` (рис. 2.8).



```
guest@aamishina:~ — /usr/bin/vim 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;
}
```

Рис. 2.8: Программа readfile

Компилируем её: `gcc readfile.c -o readfile`. Меняем владельца у файла `readfile.c` и изменяем права так, чтобы только суперпользователь (`root`) мог прочитать его, а `guest` не мог. Проверяем, что пользователь `guest` не может прочитать файл `readfile.c` (рис. 2.9).



```
[guest@aamishina ~]$ gcc readfile.c -o readfile
[guest@aamishina ~]$ sudo chown root:guest /home/guest/readfile.c

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for guest:
guest is not in the sudoers file. This incident will be reported.
[guest@aamishina ~]$ su -
Password:
Last login: Thu Apr 11 12:53:04 MSK 2024 on pts/0
[root@aamishina ~]# sudo chown root:guest /home/guest/readfile.c
[root@aamishina ~]# chmod 700 /home/guest/readfile.c
[root@aamishina ~]# exit
logout
[guest@aamishina ~]$ cat readfile.c
cat: readfile.c: Permission denied
[guest@aamishina ~]$
```

Рис. 2.9: Изменение прав доступа, проверка от имени пользователя `guest`

Меняем у программы `readfile` владельца и становим SetUID-бит. Проверяем, может ли программа `readfile` прочитать файл `readfile.c` (может). Проверяем, может ли программа `readfile` прочитать файл `/etc/shadow` (может) (рис. 2.10).

```

[guest@aamishina ~]$ su -
Password:
Last login: Thu Apr 11 12:57:53 MSK 2024 on pts/0
[root@aamishina ~]# chown root:guest /home/guest/readfile
[root@aamishina ~]# chmod u+s /home/guest/readfile
[root@aamishina ~]# exit
logout
[guest@aamishina ~]$ ./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));
        for (i = 0; i < bytes_read; ++i) printf("%c", buffer[i]);
    }
    while (bytes_read == sizeof (buffer));
    close (fd);
    return 0;
}
[guest@aamishina ~]$ ./readfile /etc/shadow
root:$6$LuGM1r0d8HjKR2ls$dIcrcF72yVYb6VKbkHgc8L0d2X5kzVRk4ca3AWsVFEZFZmHNCroIEUtruE4GQjZCt/GTqAPLE
/hSh9lIFC6Xm0::0:99999:7:::
bin:!:19469:0:99999:7:::
daemon:!:19469:0:99999:7:::
adm:!:19469:0:99999:7:::
lp:!:19469:0:99999:7:::
sync:!:19469:0:99999:7:::
shutdown:!:19469:0:99999:7:::
halt:!:19469:0:99999:7:::
mail:!:19469:0:99999:7:::
operator:!:19469:0:99999:7:::
games:!:19469:0:99999:7:::
ftp:!:19469:0:99999:7:::
nobody:!:19469:0:99999:7:::
systemd-coredump:!!:19769:!!!!:
dbus:!!:19769:!!!!:
polkitd:!!:19769:!!!!:
avahi:!!:19769:!!!!:

```

Рис. 2.10: Установка SETUID для readfile, проверка

2.2 Исследование Sticky-бита

Выясним, установлен ли атрибут Sticky на директории /tmp, для чего выполните команду `ls -l / | grep tmp`. От имени пользователя guest создаем файл file01.txt в директории /tmp со словом test: `echo "test" > /tmp/file01.txt`. Просматриваем атрибуты у только что созданного файла и разрешаем чтение и запись для категории пользователей «все остальные»: `ls -l /tmp/file01.txt`, `chmod o+rw /tmp/file01.txt` и `ls -l /tmp/file01.txt`. От пользователя guest2 (не являющегося владельцем) пробуем прочитать файл /tmp/file01.txt: `cat /tmp/file01.txt` (успешно), дозаписать в файл слово test2 командой `echo "test2" > /tmp/file01.txt` (отказано в доступе), записать в файл слово test3, стерев при этом всю имеющуюся в файле

информацию: `echo "test3" > /tmp/file01.txt` (отказано в доступе). Проверяем содержимое файла: `cat /tmp/file01.txt` (не изменилось). Попробуем удалить файл: `rm /tmp/file01.txt` (отказано в доступе) (рис. 2.11).

```
[guest@aamishina ~]$ ls -l / | grep tmp
drwxrwxrwt. 17 root root 4096 Apr 11 13:03 tmp
[guest@aamishina ~]$ echo "test" > /tmp/file01.txt
[guest@aamishina ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 Apr 11 13:06 /tmp/file01.txt
[guest@aamishina ~]$ chmod o+rw /tmp/file01.txt
[guest@aamishina ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 Apr 11 13:06 /tmp/file01.txt
[guest@aamishina ~]$ su guest2
Password:
[guest2@aamishina guest]$ cat /tmp/file01.txt
test
[guest2@aamishina guest]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@aamishina guest]$ cat /tmp/file01.txt
test
[guest2@aamishina guest]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@aamishina guest]$ cat /tmp/file01.txt
test
[guest2@aamishina guest]$ echo "test3" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@aamishina guest]$ cat /tmp/file01.txt
test
[guest2@aamishina guest]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': No such file or directory
[guest2@aamishina guest]$ cd
[guest2@aamishina ~]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': No such file or directory
[guest2@aamishina ~]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest2@aamishina ~]$
```

Рис. 2.11: Проверка наличия атрибута Sticky на /tmp, работа с файлом file01.txt

Повышаем свои права до суперпользователя следующей командой `su -` и выполняем после этого команду, снимающую атрибут `t` (Sticky-бит) с директории /tmp: `chmod -t /tmp`. Выходим из режима суперпользователя: `exit`. От пользователя `guest2` проверяем, что атрибута `t` у директории /tmp нет: `ls -l / | grep tmp`. Повторяем предыдущие шаги. Чтение и удаления удастся выполнить, на запись и перезапись получаем отказ (рис. 2.12).

```

[guest2@aamishina ~]$ su -
Password:
Last login: Thu Apr 11 13:03:10 MSK 2024 on pts/0
[root@aamishina ~]# chmod -t /tmp
[root@aamishina ~]# exit
logout
[guest2@aamishina ~]$ ls -l / | grep tmp
drwxrwxrwx. 18 root root 4096 Apr 11 13:09 tmp
[guest2@aamishina ~]$ cat /tmp/file01.txt
test
[guest2@aamishina ~]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@aamishina ~]$ echo "test3" >> /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@aamishina ~]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y

```

Рис. 2.12: Снятие атрибута t (Sticky-бит), повторение операций

Повышаем свои права до суперпользователя и возвращаем атрибут t на директорию /tmp: su -, chmod +t /tmp и exit (рис. 2.13).

```

[guest2@aamishina ~]$
[guest2@aamishina ~]$ ls -l /tmp | grep file01
[guest2@aamishina ~]$
[guest2@aamishina ~]$
[guest2@aamishina ~]$ su -
Password:
Last login: Thu Apr 11 13:09:57 MSK 2024 on pts/0
[root@aamishina ~]# chmod +t /tmp
[root@aamishina ~]# exit
logout
[guest2@aamishina ~]$ █

```

Рис. 2.13: Возвращение атрибута

3 Выводы

В ходе выполнения данной лабораторной работы, я изучила механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. 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

1. Кулябов Д.С., Королькова А.В., Геворкян М.Н. Информационная безопасность компьютерных сетей. Лабораторные работы, учебное пособие. Москва: РУДН, 2015. 64 с.