

Отчёт по лабораторной работе №5

Дисциплина: Основы информационной безопасности

Полиенко Анастасия Николаевна, НПМбд-01-19

Содержание

1	Цель работы	4
2	Теоретическое введение	5
3	Выполнение лабораторной работы	6
3.1	Создание программы	6
3.2	Исследование Sticky-бита	10
4	Выводы	14
	Список литературы	15

Список иллюстраций

3.1	Текст программы simpleid.c	6
3.2	Компиляция и запуск simpleid	7
3.3	Текст программы simpleid2.c	7
3.4	Компиляция и запуск simpleid2	7
3.5	Смена владельца и установка SetUID	8
3.6	Запуск simpleid2	8
3.7	SetGID-бит	8
3.8	Текст программы readfile.c	9
3.9	Компиляция readfile.c	9
3.10	Запуск readfile	10
3.11	Создание файла file01.txt	11
3.12	Действия над file01.txt от лица guest2	11
3.13	Удаление Sticky-бита	12
3.14	Повтор действий	12
3.15	Возвращение Sticky-бита	13

1 Цель работы

Изучить особенности работы с дополнительными атрибутами SetUID, SetGID и Sticky битами и их влияние на работу с файлами при их наличии и отсутствии.

2 Теоретическое введение

SetUID, SetGID и Sticky — это специальные типы разрешений, которые позволяют задавать расширенные права доступа на файлы и каталоги.

- SetUID — это бит разрешения, который позволяет пользователю запускать исполняемый файл с правами владельца этого файла. Другими словами, использование этого бита позволят поднять привилегии пользователя в случае, если это необходимо. Наличие SetUID бита выражается в том, что на месте классического бита x выставлен специальный бит s: `-rwsr-xr-x`
- SetGID — очень похож на SetUID с отличием, что файл будет запускаться от имени группы, который владеет файлом: `-rwxr-sr-x`
- Sticky — в случае, если этот бит установлен для папки, то файлы в этой папке могут быть удалены только их владельцем. Наличие этого бита показывается через букву t в конце всех прав: `drwxrwxrwx`

Более подробно см. в [1].

3 Выполнение лабораторной работы

3.1 Создание программы

Создадим программу simpleid.c (рис. 3.1).

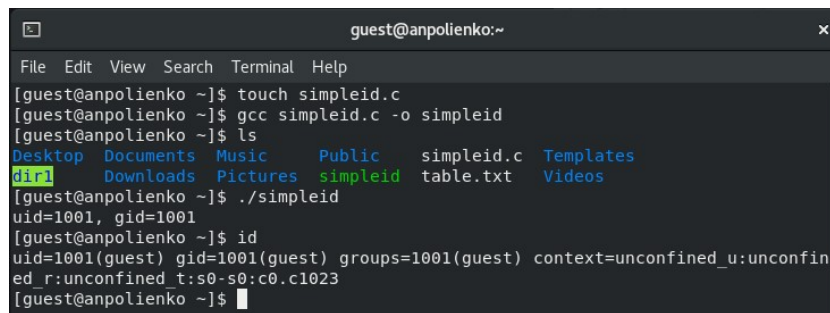


```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int
main ()
{
    uid_t uid = geteuid();
    gid_t gid = getegid();
    printf("uid=%d, gid=%d\n", uid, gid);
    return 0;
}
```

Рис. 3.1: Текст программы simpleid.c

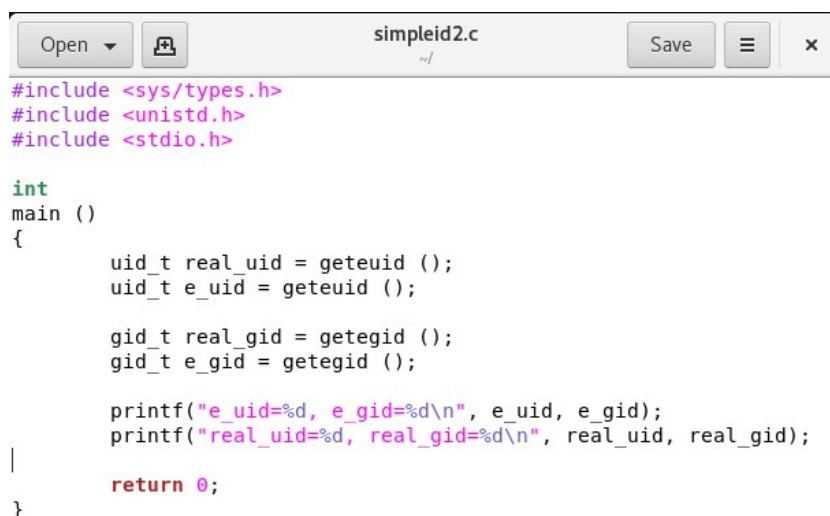
Скомпилируем программу с помощью команды gcc и убеждаемся, что файл действительно создан. Далее запускаем исполняемый файл через ./ . Вывод написанной программы совпадает с выводом команды id (рис 3.2).



```
guest@anpolienko:~  
File Edit View Search Terminal Help  
[guest@anpolienko ~]$ touch simpleid.c  
[guest@anpolienko ~]$ gcc simpleid.c -o simpleid  
[guest@anpolienko ~]$ ls  
Desktop Documents Music Public simpleid.c Templates  
Downloads Pictures simpleid table.txt Videos  
[guest@anpolienko ~]$ ./simpleid  
uid=1001, gid=1001  
[guest@anpolienko ~]$ id  
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023  
[guest@anpolienko ~]$
```

Рис. 3.2: Компиляция и запуск simpleid

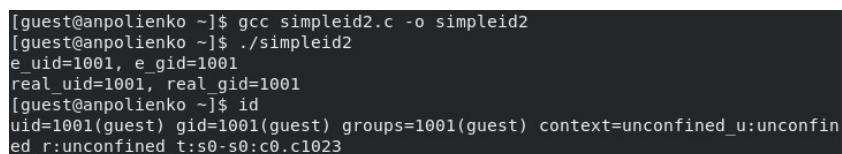
Усложним программу и назовём её simpleid2.c (рис. 3.3).



```
simpleid2.c  
#include <sys/types.h>  
#include <unistd.h>  
#include <stdio.h>  
  
int  
main ()  
{  
    uid_t real_uid = geteuid ();  
    uid_t e_uid = geteuid ();  
  
    gid_t real_gid = getegid ();  
    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. 3.3: Текст программы simpleid2.c

Скомпилируем и запустим файл simpleid2 (рис. 3.4).



```
[guest@anpolienko ~]$ gcc simpleid2.c -o simpleid2  
[guest@anpolienko ~]$ ./simpleid2  
e_uid=1001, e_gid=1001  
real uid=1001, real gid=1001  
[guest@anpolienko ~]$ id  
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 3.4: Компиляция и запуск simpleid2

От имени суперпользователя сменим владельца файла simpleid2 на root и установим SetUID-бит. Далее через команду `ls -l` видим, что бит установлен корректно (рис. 3.5)

```
[anpolienko@anpolienko ~]$ sudo chown root:guest /home/guest/simpleid2
[sudo] password for anpolienko:
[anpolienko@anpolienko ~]$ sudo chmod u+s /home/guest/simpleid2
[anpolienko@anpolienko ~]$ ls -l /home/guest/simpleid2
ls: cannot access '/home/guest/simpleid2': Permission denied
[anpolienko@anpolienko ~]$ sudo ls -l /home/guest/simpleid2
-rwsrwxr-x. 1 root guest 18152 Sep 21 01:09 /home/guest/simpleid2
```

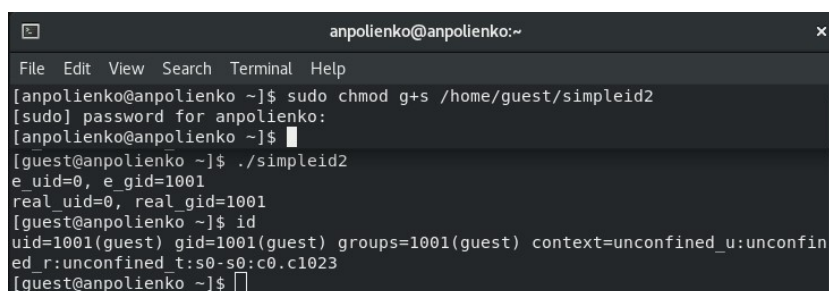
Рис. 3.5: Смена владельца и установка SetUID

Запускаем программу simpleid2 и команду id. Теперь видим, что появились отличия в uid строках (рис. 3.6).

```
[guest@anpolienko ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=1001
[guest@anpolienko ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@anpolienko ~]$
```

Рис. 3.6: Запуск simpleid2

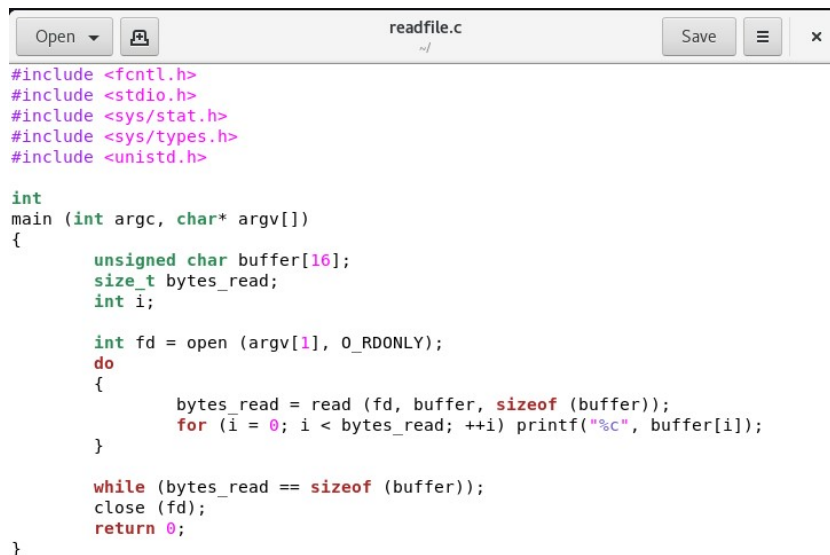
Продолжаем выше описанные действия для SetGID-бита. Теперь после запуска simpleid2 можем увидеть отличие и в gid строках (рис. 3.7).



```
anpolienko@anpolienko:~
File Edit View Search Terminal Help
[anpolienko@anpolienko ~]$ sudo chmod g+s /home/guest/simpleid2
[sudo] password for anpolienko:
[anpolienko@anpolienko ~]$
[guest@anpolienko ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=1001
[guest@anpolienko ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@anpolienko ~]$
```

Рис. 3.7: SetGID-бит

Создадим программу readfile.c (рис. 3.8).



```

#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;

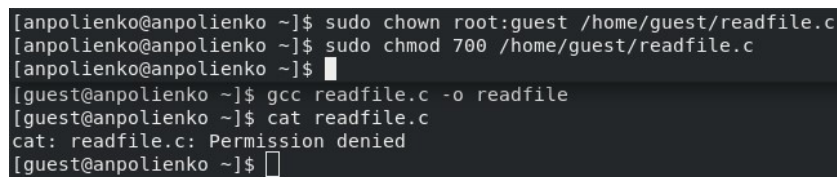
    int fd = open (argv[1], O_RDONLY);
    do
    {
        bytes_read = read (fd, buffer, sizeof (buffer));
        for (i = 0; i < bytes_read; ++i) printf ("%c", buffer[i]);
    }

    while (bytes_read == sizeof (buffer));
    close (fd);
    return 0;
}

```

Рис. 3.8: Текст программы readfile.c

Откомпилируем эту программу командой gcc. Далее меняем владельца файла readfile.c и отнимаем у пользователя guest право на чтение. При попытке прочитать файл от имени пользователя guest возникает ошибка (рис. 3.9).



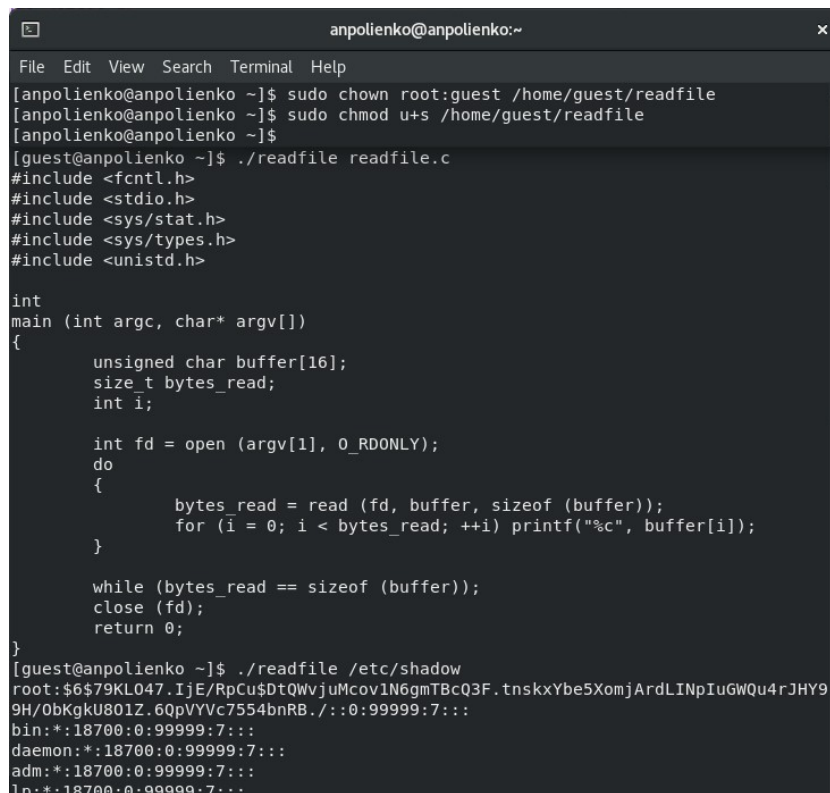
```

[anpolienko@anpolienko ~]$ sudo chown root:guest /home/guest/readfile.c
[anpolienko@anpolienko ~]$ sudo chmod 700 /home/guest/readfile.c
[anpolienko@anpolienko ~]$ 
[guest@anpolienko ~]$ gcc readfile.c -o readfile
[guest@anpolienko ~]$ cat readfile.c
cat: readfile.c: Permission denied
[guest@anpolienko ~]$ 

```

Рис. 3.9: Компиляция readfile.c

Меняем владельца файла readfile и устанавливаем на него SetUID-бит. Запускаем исполняемый файл и убеждаемся, что программа может прочитать файлы readfile.c и /etc/shadow (рис. 3.10).



```
anpolienko@anpolienko:~  
File Edit View Search Terminal Help  
[anpolienko@anpolienko ~]$ sudo chown root:guest /home/guest/readfile  
[anpolienko@anpolienko ~]$ sudo chmod u+s /home/guest/readfile  
[anpolienko@anpolienko ~]$  
[guest@anpolienko ~]$ ./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@anpolienko ~]$ ./readfile /etc/shadow  
root:$6$79KL047.IjE/RpCu$DtQWvjuMcov1N6gmTBcQ3F.tnskxYbe5XomjArdLINpIuGwQu4rJHY9  
9H/ObKgkU801Z.6QpVYVc7554bnRB./::0:99999:7:::  
bin:*.18700:0:99999:7:::  
daemon:*.18700:0:99999:7:::  
adm:*.18700:0:99999:7:::  
lp:*.18700:0:99999:7:::
```

Рис. 3.10: Запуск readfile

3.2 Исследование Sticky-бита

Выполняя команду `ls -l` выявляем, что на каталоге `/tmp` установлен Sticky-бит. Это видно, т.к. в конце написана `t`. Далее от имени пользователя `guest` создаём файл `/tmp/file01.txt`. Потом просматриваем атрибуты только что созданного файла и даём всем пользователям право на чтение и запись (рис. 3.11).

```
[anpolienko@anpolienko ~]$ ls -l / | grep tmp
drwxrwxrwt. 18 root root 4096 Sep 22 00:27 tmp

guest@anpolienko:~
File Edit View Search Terminal Help
[anpolienko@anpolienko ~]$ su - guest
Password:
[guest@anpolienko ~]$ echo " test" > /tmp/file01.txt
[guest@anpolienko ~]$ ls -l /tmp/file01.txt
-rw-rw-r--. 1 guest guest 6 Sep 22 00:30 /tmp/file01.txt
[guest@anpolienko ~]$ chmod o+rw /tmp/file01.txt
[guest@anpolienko ~]$ ls -l /tmp/file01.txt
-rw-rw-rw-. 1 guest guest 6 Sep 22 00:30 /tmp/file01.txt
[guest@anpolienko ~]$
```

Рис. 3.11: Создание файла file01.txt

От имени пользователя guest2 читаем файл file01.txt командой cat. Далее успешно дозаписываем в конец файла строку “test2”, а затем успешно перезаписываем содержимое, меняя его на строку “test3”. Однако при попытке удалить файл возникла ошибка (рис. 3.12).

```
guest2@anpolienko:~
File Edit View Search Terminal Help
[anpolienko@anpolienko ~]$ su - guest2
Password:
[guest2@anpolienko ~]$ cat /tmp/file01.txt
 test
[guest2@anpolienko ~]$ echo "test2" >> /tmp/file01.txt
[guest2@anpolienko ~]$ cat /tmp/file01.txt
 test
test2
[guest2@anpolienko ~]$ echo "test3" > /tmp/file01.txt
[guest2@anpolienko ~]$ cat /tmp/file01.txt
test3
[guest2@anpolienko ~]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest2@anpolienko ~]$
```

Рис. 3.12: Действия над file01.txt от лица guest2

Временно повышаем права до суперпользователя и снимаем с директории /tmp Sticky-бит. Покидаем режим суперпользователя командой exit (рис. 3.13).

```
[anpolienko@anpolienko ~]$ su -
Password:
[root@anpolienko ~]# chmod -t /tmp
[root@anpolienko ~]# exit
logout
[anpolienko@anpolienko ~]$ █
```

Рис. 3.13: Удаление Sticky-бита

Убеждаемся через команду `ls -l`, что Sticky-бит действительно отсутствует. Далее повторяем действия от имени пользователя `guest2`, описанные выше. В этот раз удалось удалить файл `file01.txt` даже при условии, что `guest2` не является его владельцем (рис. 3.14).

```
[guest2@anpolienko ~]$ ls -l / | grep tmp
drwxrwxrwx. 18 root root 4096 Sep 22 00:41 tmp
[guest2@anpolienko ~]$ cat /tmp/file01.txt
test3
[guest2@anpolienko ~]$ echo "test2" >> /tmp/file01.txt
[guest2@anpolienko ~]$ cat /tmp/file01.txt
test3
test2
[guest2@anpolienko ~]$ echo "test3" > /tmp/file01.txt
[guest2@anpolienko ~]$ cat /tmp/file01.txt
test3
[guest2@anpolienko ~]$ rm /tmp/file01.txt
[guest2@anpolienko ~]$ ls /tmp | grep *.txt
[guest2@anpolienko ~]$ ls /tmp | grep file01.txt
[guest2@anpolienko ~]$ █
```

Рис. 3.14: Повтор действий

Временно повышаем права до суперпользователя и возвращает Sticky-бит на каталог `/tmp` (рис. 3.15).

```
[anpolienko@anpolienko ~]$ su -  
Password:  
[root@anpolienko ~]# chmod +t /tmp  
[root@anpolienko ~]# exit  
logout  
[anpolienko@anpolienko ~]$ ls -l / | grep tmp  
drwxrwxrwt. 18 root root 4096 Sep 22 00:46 tmp  
[anpolienko@anpolienko ~]$
```

Рис. 3.15: Возращение Sticky-бита

4 Выводы

Изучила механизмы изменения идентификаторов и получила практические навыки по работе с SetUID, SetGID и Sticky битами и узнала об их особенностях и влиянии на файлы и директории.

Список литературы

1. SETUID, SETGID and Sticky bits [Электронный ресурс]. RuVSD, 2021. URL: <https://ruvsd.com/ru/helpcenter/suid-sgid-sticky-bit-linux/>.