

Лабораторная работа №5

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Маслова Анастасия Сергеевна

Содержание

1	Цель работы	1
2	Выполнение лабораторной работы.....	1
2.1	Создание программы	1
2.2	Исследование Sticky-бита.....	6
3	Выводы	7
	Список литературы	7

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 Выполнение лабораторной работы

2.1 Создание программы

1. Войдите в систему от имени пользователя guest.
2. Создайте программу 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;
}
```

Выполнение этих двух пунктов вы можете видеть ниже (рис. [??]).

```
guest@asmaslova:~/Documents
File Edit View Search Terminal Help
[guest@asmaslova Documents]$ touch simpleid.c
[guest@asmaslova Documents]$ cat > simpleid.c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int main(){
    uid_t uid = getuid ();
    gid_t gid = getegid ();
    printf ("uid=%d, gid=%d\n", uid, gid);
    return 0;
}
[guest@asmaslova Documents]$ cat simpleid.c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int main(){
    uid_t uid = getuid ();
    gid_t gid = getegid ();
    printf ("uid=%d, gid=%d\n", uid, gid);
    return 0;
}
[guest@asmaslova Documents]$
```

Создание программы *simpleid.c*

3. Скомпилируйте программу и убедитесь, что файл программы создан: `gcc simpleid.c -o simpleid`
4. Выполните программу *simpleid*: `./simpleid`
5. Выполните системную программу *id*: `id` и сравните полученный вами результат с данными предыдущего пункта задания.

Выполнение этих трех пунктов вы можете видеть ниже (рис. [??]).

```
[guest@asmaslova Documents]$ gcc simpleid.c -o simpleid
[guest@asmaslova Documents]$ gcc simpleid.c -o simpleid
[guest@asmaslova Documents]$ ls
dir1 simpleid simpleid.c
[guest@asmaslova Documents]$ rm simpleid
[guest@asmaslova Documents]$ ls
dir1 simpleid simpleid.c
[guest@asmaslova Documents]$ ./simpleid
uid=1001, gid=1001
[guest@asmaslova Documents]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@asmaslova Documents]$
```

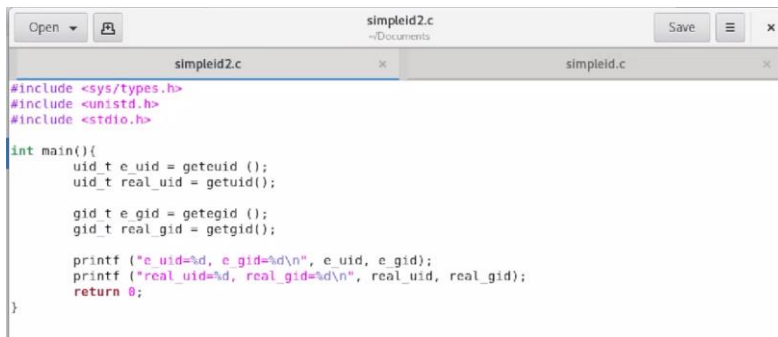
Компиляция и выполнение программы *simpleid.c*

6. Усложните программу, добавив вывод действительных идентификаторов:

```
#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;
}
```

Получившуюся программу назовите simpleid2.c.

Я создала файл simpleid2.c и записала туда этот код (рис. [??]).



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

int main(){
    uid_t e_uid = geteuid ();
    uid_t real_uid = getuid();

    gid_t e_gid = getegid ();
    gid_t real_gid = getgid();

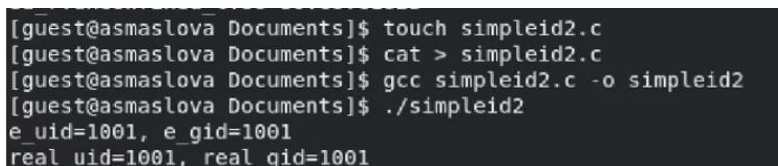
    printf ("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
    printf ("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
    return 0;
}
```

Создание программы simpleid2.c

7. Скомпилируйте и запустите simpleid2.c:

```
gcc simpleid2.c -o simpleid2
./simpleid2
```

Я скомпилировала и запустила названный файл (рис. [??]).



```
[guest@asmaslova Documents]$ touch simpleid2.c
[guest@asmaslova Documents]$ cat > simpleid2.c
[guest@asmaslova Documents]$ gcc simpleid2.c -o simpleid2
[guest@asmaslova Documents]$ ./simpleid2
e_uid=1001, e_gid=1001
real uid=1001, real gid=1001
```

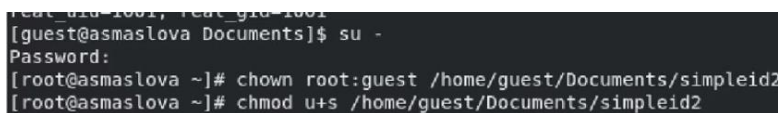
Компиляция и выполнение программы simpleid2.c

8. От имени суперпользователя выполните команды:

```
chown root:guest /home/guest/simpleid2
chmod u+s /home/guest/simpleid2
```

9. Используйте sudo или повысьте временно свои права с помощью su. Поясните, что делают эти команды.

Повысив свои права с помощью команды su -, я от имени суперпользователя выполнила указанные команды для изменения прав на файл simpleid2 (рис. [??]).



```
real uid=1001, real gid=1001
[guest@asmaslova Documents]$ su -
Password:
[root@asmaslova ~]# chown root:guest /home/guest/Documents/simpleid2
[root@asmaslova ~]# chmod u+s /home/guest/Documents/simpleid2
```

Изменение прав на файл simpleid2 от имени суперпользователя

10. Выполните проверку правильности установки новых атрибутов и смены владельца файла simpleid2:

```
ls -l simpleid2
```

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

```
[guest@asmaslova Documents]$ ls -l
total 48
drwxrwxr-x. 2 guest guest 33 Sep 28 11:22 dirl
-rwxrwxr-x. 1 guest guest 18208 Oct 5 07:45 simpleid
-rwsrwxr-x. 1 root guest 18312 Oct 5 07:52 simpleid2
-rw-rw-r--. 1 guest guest 308 Oct 5 07:52 simpleid2.c
-rw-rw-r--. 1 guest guest 178 Oct 5 07:49 simpleid.c
```

Проверка правильности установки новых атрибутов

11. Запустите simpleid2 и id:
`./simpleid2`
`id`

Сравните результаты.

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

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

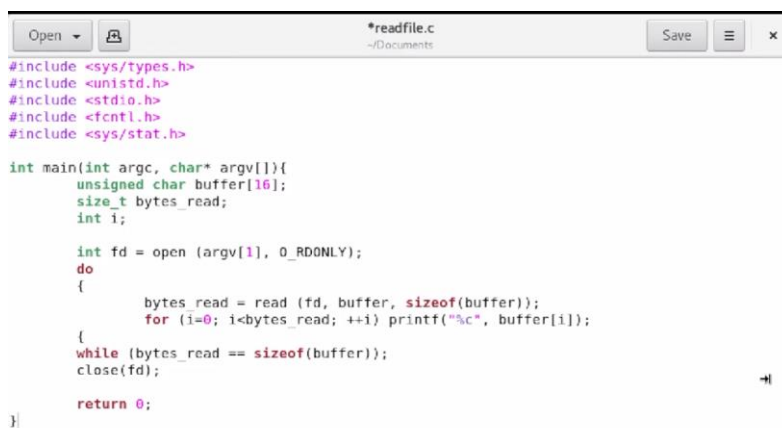
Компиляция и выполнение программы simpleid2.c

По сравнению с работой программы simpleid, в выводе simpleid2 присутствовали отличающиеся данные, а конкретно - e_uid=0.

12. Прodelайте тоже самое относительно SetGID-бита.
13. Создайте 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;
}
```

Выполнение этого пункта вы можете видеть ниже (рис. [??]).



```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>
#include <sys/stat.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);

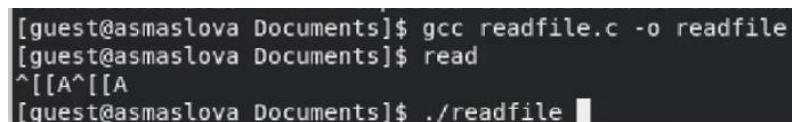
    return 0;
}
```

Создание программы readfile.c

14. Откомпилируйте её.

`gcc readfile.c -o readfile`

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

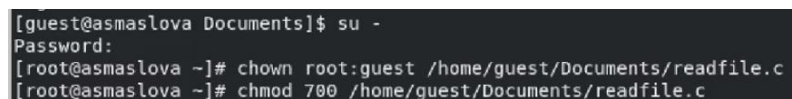


```
[guest@asmaslova Documents]$ gcc readfile.c -o readfile
[guest@asmaslova Documents]$ read
^[[A^[[A
[guest@asmaslova Documents]$ ./readfile
```

Компиляция и выполнение программы readfile.c

15. Смените владельца у файла readfile.c (или любого другого текстового файла в системе) и измените права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог.

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

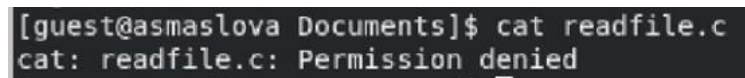


```
[guest@asmaslova Documents]$ su -
Password:
[root@asmaslova ~]# chown root:guest /home/guest/Documents/readfile.c
[root@asmaslova ~]# chmod 700 /home/guest/Documents/readfile.c
```

Смена владельца файла readfile.c

16. Проверьте, что пользователь guest не может прочитать файл readfile.c.

Выполнение этого пункта вы можете видеть ниже (рис. [??]).



```
[guest@asmaslova Documents]$ cat readfile.c
cat: readfile.c: Permission denied
```

Невозможность прочитать файл readfile.c

17. Смените у программы readfile владельца и установите SetU'D-бит.

18. Проверьте, может ли программа readfile прочитать файл readfile.c?

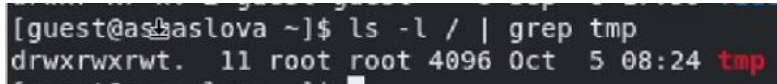
19. Проверьте, может ли программа readfile прочитать файл /etc/shadow? Отразите полученный результат и ваши объяснения в отчёте.

2.2 Исследование Sticky-бита

1. Выясните, установлен ли атрибут Sticky на директории /tmp, для чего выполните команду

```
ls -l / | grep tmp
```

Выполнение этого пункта вы можете видеть ниже (рис. [??]). Как можно заметить, атрибут установлен, потому что в конце стоит буква t.



```
[guest@asdaslova ~]$ ls -l / | grep tmp
drwxrwxrwt. 11 root root 4096 Oct 5 08:24 tmp
```

Проверка наличия атрибута Sticky на директории /tmp

2. От имени пользователя guest создайте файл file01.txt в директории /tmp со словом test:

```
echo "test" > /tmp/file01.txt
```

3. Просмотрите атрибуты у только что созданного файла и разрешите чтение и запись для категории пользователей «все остальные»:

```
ls -l /tmp/file01.txt
chmod o+rw /tmp/file01.txt
ls -l /tmp/file01.txt
```

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

Просмотр атрибутов

Просмотр атрибутов

4. От пользователя guest2 (не являющегося владельцем) попробуйте прочитать файл /tmp/file01.txt: `cat /tmp/file01.txt`
5. От пользователя guest2 попробуйте дозаписать в файл /tmp/file01.txt слово test2 командой `echo "test2" > /tmp/file01.txt`. Удалось ли вам выполнить операцию?

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

Попытка дозаписать в файл

Попытка дозаписать в файл

6. Проверьте содержимое файла командой `cat /tmp/file01.txt`

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

Проверка содержимого файла

Проверка содержимого файла

7. От пользователя guest2 попробуйте записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой `echo "test3" > /tmp/file01.txt`. Удалось ли вам выполнить операцию?
8. Проверьте содержимое файла командой `cat /tmp/file01.txt`
9. От пользователя guest2 попробуйте удалить файл /tmp/file01.txt командой `rm /tmp/file01.txt`. Удалось ли вам удалить файл?

10. Повысьте свои права до суперпользователя следующей командой `su -` и выполните после этого команду, снимающую атрибут `t` (Sticky-бит) с директории `/tmp`: `chmod -t /tmp`
11. Покиньте режим суперпользователя командой `exit`
12. От пользователя `guest2` проверьте, что атрибута `t` у директории `/tmp` нет: `ls -l / | grep tmp`

Выполнение этого пункта вы можете видеть ниже (рис. [??]).

Проверка отсутствия атрибута Sticky на директории `/tmp`

Проверка отсутствия атрибута Sticky на директории `/tmp`

13. Повторите предыдущие шаги. Какие наблюдаются изменения?

Единственное изменение, которое я заметила - это тот факт, что с этим атрибутом я могу удалить файл `readfile.c`.

14. Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем? Ваши наблюдения занесите в отчёт.
15. Повысьте свои права до суперпользователя и верните атрибут `t` на директорию `/tmp`:

```
su -  
chmod +t /tmp  
exit
```

3 Выводы

В ходе лабораторной работы я изучила механизмы изменения идентификаторов, применения SetUID- и Sticky-битов, получила практические навыки работы в консоли с дополнительными атрибутами, рассмотрела работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

Список литературы