

Отчёт по лабораторной работе 5

Агеева Анастасия Борисовна

12 ноября, 2021

Изучение механизмов изменения идентификаторов, применения SetUID- и Sticky-битов. Получение практических навыков работы в кон- соли с дополнительными атрибутами. Рассмотрение работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

Лабораторная работа подразумевает создание программы и использование Sticky-бита.

Выполнение лабораторной работы

1. Войдем в систему от имени пользователя guest. (рис.1).

2. Создаем программу simpleid.c (рис.1).

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

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

Figure 1: рис.1. Программа simpleid.c.

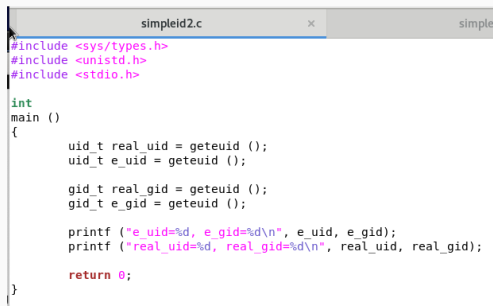
3. Скомпилируем программу и убедимся, что файл программы создан (рис.3-6).
4. Выполним программу simpleid (рис.7).

5. Выполним системную программу `id` и сравним полученный нами результат с данными предыдущего пункта задания (рис.8).

```
[guest@abageeval ~]$ gcc simpleid.c -o simpleid
[guest@abageeval ~]$ ./simpleid
uid=1001, gid=1001
[guest@abageeval ~]$ id
uid=1001(guest) gid=1001(guest) rpnны=1001(guest),10(wheel) контекст=unconfined_u:unco
nfinеd_r:unconfined_t:s0-s0:c0.c1023
[guest@abageeval ~]$ █
```

Figure 2: рис.2. Компиляция и выполнение программы.

6. Усложним программу, добавив вывод действительных идентификаторов. Получившуюся программу назовем simpleid2.c (рис.9).



```
simpleid2.c  ×  simplei
#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 = geteuid ();
    gid_t e_gid = geteuid ();

    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;
}
```

Figure 3: рис.3. Программа simpleid2.c.

7. Скомпилируем и запустим `simpleid2.c` (рис.10-11).
8. От имени суперпользователя выполним команды:
`chown root:guest /home/guest/simpleid2` и `chmod u+s /home/guest/simpleid2` (рис.12).
9. Повысим временно свои права с помощью `su`.
(рис.10-11).
10. Выполним проверку правильности установки новых атрибутов и смены владельца файла `simpleid2`
(рис.10-11).

11. Запустим simpleid2 и id и сравним результат (рис.10-11).

```
[guest@abageeval ~]$ nano simpleid2.c
[guest@abageeval ~]$ gcc simpleid2.c -o simpleid2
[guest@abageeval ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@abageeval ~]$ id
uid=1001(guest) gid=1001(guest) rpyнны=1001(guest),10(wheel) контекст=unconfined_u:unco
nfinеd r:unconfined t:s0-s0:c0.c1023
[guest@abageeval ~]$ su
Польз:
[root@abageeval guest]# chown root:guest /home/guest/simpleid2
[root@abageeval guest]# chmod u+s /home/guest/simpleid2
[root@abageeval guest]# ls -l simpleid2
-rwsrwxr-x. 1 root guest 8656 ноя 11 23:30 simpleid2
[root@abageeval guest]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@abageeval guest]# id
uid=0(root) gid=0(root) rpyнны=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:
s0-s0:c0.c1023
[root@abageeval guest]# █
```

Figure 4: рис.4. Работа с программой simpleid2.c.

12. Проделаем тоже самое относительно SetGID-бита (рис.10-11).

```
[root@abageeval guest]# chmod g+s /home/guest/simpleid2
[root@abageeval guest]# ls -l simpleid2
-rwsrwsr-x. 1 root guest 8656 ноя 11 23:30 simpleid2
[root@abageeval guest]# ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=0
[root@abageeval guest]# id I
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:
s0-s0:c0.c1023
[root@abageeval guest]# █
```

Figure 5: рис.5. Действия относительно SetGID-бита.

13. Создаем программу readfile.c (рис.10-11).

```
#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) print("%c", buffer[i]);
    }

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

Figure 6: рис.6. Программа readfile.c.

14. Откомпилируем её (рис.10-11).
15. Сменим владельца у файла readfile.c (или любого другого текстового файла в системе) и изменим права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог (рис.10-11).
16. Проверим, что пользователь guest не может прочитать файл readfile.c (рис.10-11).

17. Сменим у программы readfile владельца и установим SetU'D-бит (рис.10-11).

```
[root@abageeval guest]# gcc readfile.c -o readfile
[root@abageeval guest]# chmod a-r readfile.c
[root@abageeval guest]# exit
exit
[guest@abageeval ~]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@abageeval ~]$ su
Пароль:
[root@abageeval guest]# chown root:guest readfile
[root@abageeval guest]# chmod u+s readfile
[root@abageeval guest]# exit
exit
[guest@abageeval ~]$ █
```

Figure 7: рис.7. Работа с программой readfile.c.

18. Проверим, может ли программа readfile прочитать файл readfile.c (рис.10-11).

```
[guest@abageeval ~]$ ./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@abageeval ~]$ █
```

Figure 8: рис.8. Программа readfile читает файл readfile.c.

19. Проверим, может ли программа readfile прочитать файл /etc/shadow (рис.10-11).

```
[guest@abageeval ~]$ ./readfile /etc/shadow
root:$6xzzkG4rn]RBS.c.dE$kj.rIs2Eh4/jZ6lqe.u0b6/ngdXdsETCeMwsQerJU5LByPFm0QY7uGdUjYYK0mP
A6sveVqrCSUyyrrEWOAIJ05.::0:99999:7:::
bin:!:17834:0:99999:7:::
daemon:!:17834:0:99999:7:::
adm:!:17834:0:99999:7:::
lp:!:17834:0:99999:7:::
sync:!:17834:0:99999:7:::
shutdown:!:17834:0:99999:7:::
halt:!:17834:0:99999:7:::
mail:!:17834:0:99999:7:::
operator:!:17834:0:99999:7:::
games:!:17834:0:99999:7:::
ftp:!:17834:0:99999:7:::
nobody:!:17834:0:99999:7:::
systemd-network:!!:18900::::::
dbus:!!:18900::::::
polkitd:!!:18900::::::
libstoragemgmt:!!:18900::::::
colord:!!:18900::::::
rpc:!!:18900:0:99999:7:::
saslauth:!!:18900::::::
abrt:!!:18900::::::
rtkit:!!:18900::::::
pulse:!!:18900::::::
cadvd:!!:18900::::::
```

Figure 9: рис.9. Программа readfile читает файл /etc/shadow

20. Выясним, установлен ли атрибут Sticky на директории /tmp (рис.10-11).
21. От имени пользователя guest создадим файл file01.txt в директории /tmp со словом test (рис.10-11).

22. Посмотрим атрибуты у только что созданного файла и разрешим чтение и запись для категории пользователей «все остальные» (рис.10-11).

```
[guest@abageeval ~]$ ls -l / | grep tmp
drwxrwxrwt. 16 root root 4096 ноя 11 23:49 tmp
[guest@abageeval ~]$ echo "test" > /tmp/file01.txt
[guest@abageeval ~]$ ls -l /tmp/file01.txt
-rw-rw-r--. 1 guest guest 5 ноя 11 23:52 /tmp/file01.txt
[guest@abageeval ~]$ chmod o+rw /tmp/file01.txt
[guest@abageeval ~]$ ls -l /tmp/file01.txt
-rw-rw-rw-. 1 guest guest 5 ноя 11 23:52 /tmp/file01.txt
[guest@abageeval ~]$ █
```

Figure 10: рис.10. Атрибут Sticky.

23. От пользователя guest2 (не являющегося владельцем) попробуем прочитать файл /tmp/file01.txt (рис.10-11).
24. От пользователя guest2 попробуем дозаписать в файл /tmp/file01.txt слово test2 (рис.10-11).
25. Проверим содержимое файла (рис.10-11).
26. От пользователя guest2 попробуем записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию (рис.10-11).
27. Проверим содержимое файла командой (рис.10-11).
28. От пользователя guest2 попробуем удалить файл /tmp/file01.txt (рис.10-11).
29. Повысим свои права до суперпользователя командой su - и выполним после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp (рис.10-11).

33. Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем? нет (рис.10-11).

```
[guest@abageeval ~]$ su guest2
Пароль:
[guest2@abageeval guest]$ cat /tmp/file01.txt
test
[guest2@abageeval guest]$ echo "test2" > /tmp/file01.txt
[guest2@abageeval guest]$ cat /tmp/file01.txt
test2
[guest2@abageeval guest]$ echo "test3" > /tmp/file01.txt
[guest2@abageeval guest]$ cat /tmp/file01.txt
test3
[guest2@abageeval guest]$ rm /tmp/file01.txt
rm: невозможно удалить «/tmp/file01.txt»: Операция не позволена
[guest2@abageeval guest]$ su -
Пароль:
Последний вход в систему: Чт ноя 11 23:48:35 MSK 2021 на pts/1
[root@abageeval ~]# chmod -t /tmp
[root@abageeval ~]# exit
logout
[guest2@abageeval guest]$ ls -l / | grep tmp
drwxrwxrwx. 16 root root 4096 ноя 11 23:59 tmp
[guest2@abageeval guest]$ cat /tmp/file01.txt
test3
[guest2@abageeval guest]$ echo "test2" > /tmp/file01.txt
[guest2@abageeval guest]$ cat /tmp/file01.txt
test2
[guest2@abageeval guest]$ rm /tmp/file01.txt
```

Figure 11: рис.11. Работа от пользователя guest2.

34. Повысим свои права до суперпользователя и вернем атрибут `t` на директорию `/tmp` (рис.10-11).

```
[guest2@abageeval guest]$ su -  
Пароль:  
Последний вход в систему: Чт ноя 11 23:58:45 MSK 2021 на pts/1  
[root@abageeval ~]# chmod +t /tmp  
[root@abageeval ~]# exit  
logout  
[guest2@abageeval guest]$ █
```

Figure 12: рис.12. Возвращение атрибута `t` на дир `/tmp`.

Выводы

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.

Спасибо за внимание