Лабораторная работа 5

Юдин Герман Станиславович, НФИбд-01-19

Содержание

List of Figures

List of Tables

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

дисциплина: Информационная безопасность

Преподователь: Кулябов Дмитрий Сергеевич

Студент: Юдин Герман Станиславович

Группа: НФИбд-01-19

МОСКВА

2022 г.

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

Setuid – это бит разрешения, который позволяет пользователю запускать исполняемый файл с правами владельца этого файла.

Принцип работы Setgid очень похож на setuid с отличием, что файл будет запускаться пользователем от имени группы, которая владеет файлом.

Последний специальный бит разрешения – это Sticky Bit . В случае, если этот бит установлен для папки, то файлы в этой папке могут быть удалены только их владельцем.

3 Выполнение лабораторной работы

1. Создал файл simpleid.c, который скомпилировал и запустил. Выполнил команду id и сравнил вывод. Вывело одинаковую информацию (Рис fig. 3.1).

Figure 3.1: simpleid

2. Усложнил программу в файле simpleid2.c, также скомпилировал и запустил (Рис fig. 3.2). На этот раз выдаёт две идентификатора пользователя и 2 идентификатора группы.

```
[guest@gsyudin Work]$ cat simpleid2.c
#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;
}
[guest@gsyudin Work]$ gcc simpleid2.c
[guest@gsyudin Work]$ ls
a.out  readfile.c  simpleid2.c  simpleid.c
[guest@gsyudin Work]$ ./a.out
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
```

Figure 3.2: simpleid2

3. От имени суперпользователя изменил владельца файла на рута и добавил S бит на пользователя (Рис fig. 3.3). Проверил корректность установленного бита (Рис fig. 3.4).

```
[root@gsyudin ~]# chown root:guest /home/guest/Work/simpleid2
[root@gsyudin ~]# chmod u+s /home/guest/Work/simpleid2
```

Figure 3.3: change parameters

```
[guest@gsyudin Work]$ ls -l
total 68
-rwxrwxr-x. 1 guest guest 26008 Sep 27 21:40 a.out
-rwxrwx---. 1 guest guest   479 Sep 27 21:37 readfile.c
-rwsrwxr-x. 1 root  guest 26008 Sep 27 21:41 simpleid2
-rwxrwx---. 1 guest guest   345 Sep 27 21:37 simpleid2.c
-rwxrwx---. 1 guest guest   203 Sep 27 21:37 simpleid.c
```

Figure 3.4: ls -l

4. При запуске от guest выдаёт следующую информацию. Здесь мы видим, что e_uid стал 0 (Рис fig. 3.5), что означает, что файл запустился от имени владельца, то есть root.

```
[guest@gsyudin Work]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@gsyudin Work]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest),976(vboxsf) context=unconfine
d_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Figure 3.5: start simpleid2

5. Создал новый файл readfile.c, который предназначен для чтения файлов (Рис fig. 3.6). Сделал так, чтобы другой файл мог проитать root, но не мог guest (Рис fig. 3.7).

```
[guest@gsyudin Work]$ cat 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;
}
```

Figure 3.6: readfile

```
[root@gsyudin ~]# chown root:guest /home/guest/Work/readfile.c
[root@gsyudin ~]# cdmod 700 /home/guest/Work/readfile.c
bash: cdmod: command not found...
[root@gsyudin ~]# chmod 700 /home/guest/Work/readfile.c
```

Figure 3.7: change file

6. Попытался прочитать с guest, но доступ закрыт (Рис fig. 3.8 и fig. 3.9).

Figure 3.8: try read

Figure 3.9: error

7. Сменил владельца файла и установил SetU'D-бит (Рис fig. 3.10). На этот раз от пользователя guest имеется доступ ко всем файлам, к которым есть доступ у root (Рис fig. 3.11). Set биты позволяют делать действия относительно владельца файла, то есть файлы запускаются от лица владельца файла, и если владелец может прочитать какой-то файл, а тот кто запускает нет, то с помощью set бита есть возможность прочитать файл даже у пользователя не имеющего возможности прочитать данный файл.

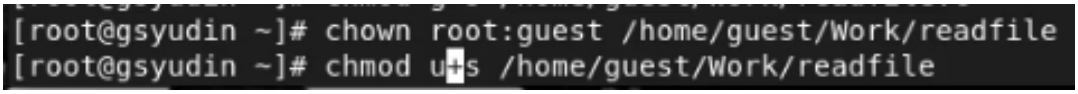

Figure 3.10: root access

```
[guest@gsyudin Work]$ ./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;
}
```

Figure 3.11: read with readfile

8. Посмотерл атрибуты на каталоге /tmp (Рис fig. 3.12).

```
[guest@gsyudin Work]$ ls -l / | grep tmp
drwxrwxrwt.  17 root root 4096 Sep 27 21:54 tmp
```

Figure 3.12: ls tmp

9. Создал в данном каталоге файл от пользователя guest (Рис fig. 3.13).

```
[guest@gsyudin Work]$ echo "test" > /tmp/file01.txt
```

Figure 3.13: echo file

10. Посмотрел атрибуты у только что созданного файла и разрешил чтение и запись для категории пользователей «все остальные» (Рис fig. 3.14).

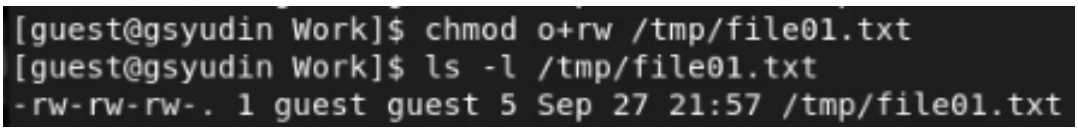

Figure 3.14: rw for other

11. От пользователя guest2 попробовал дозаписать файл. Попробовал перезаписать файл. Все действия выполнились (Рис fig. 3.15).

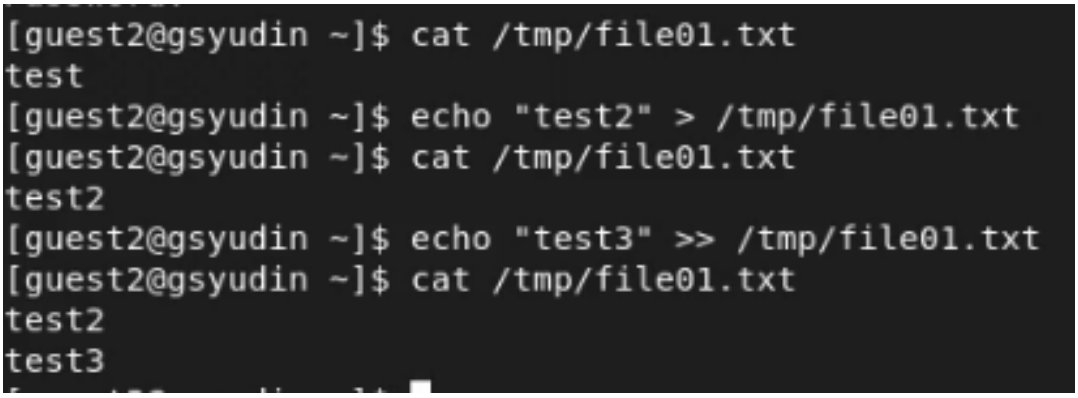

Figure 3.15: add text, change text

12. Попробовал удалить файл (Рис fig. 3.16).

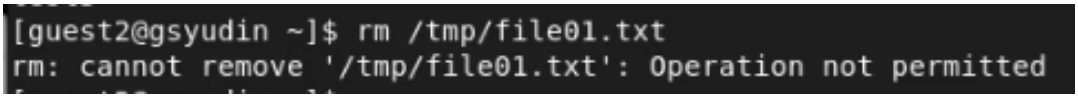

Figure 3.16: rm file fail

13. От прав администратора снял t бит с директории /tmp (Рис fig. 3.17).

Figure 3.17: -t /tmp

14. Повторил предыдущие шаги от guest2. На этот раз смог удалить файл (Рис fig. 3.18).

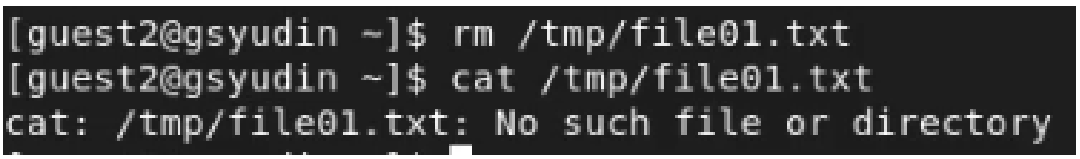

Figure 3.18: rm file success

15. Вернул t бит обратно.

4 Вывод

Выполнив данную лабораторную работу, я изучил механизмы изменения идентификаторов, применения SetUID- и Sticky-битов, получил практические навыки работы в консоли с дополнительными атрибутами, рассмотрел работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

5 Список литературы

1. Кулябов, Д.С. - Лабораторная работа № 5. Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов https://esystem.rudn.ru/pluginfile.php/1651889/mod_resource/content/2/005-lab_discret_sticky.pdf