ЛР №5. Исследование влияния дополнительных атрибутов

Дисциплина: Информационная безопасность

Доборщук Владимир Владимирович, НФИбд-01-18

13 ноября 2021

Цель работы

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.

Задачи:

- изучить механизмы изменения идентификаторов;
- укрепление навыков работы в консоли;
- изучение влияния бита Sticky на запись и удаление файлов.

Теоретическое введение

Теоретическое введение

Для выполнения данной лабораторной работы мы использовали данные источники, в виде описания лабораторной работы, а также свободные источники в интернете.

Выполнение лабораторной работы

1. Создание программы

Зайдя в терминал, мы сделали вошли в систему от лица `guest` и реализовали следующий набор скриптов:

```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. 1: `simpleid.c`

1. Создание программы

```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;
}
~
~
~
~
```

Рис. 2: `simpleid2.c`

1. Создание программы

```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. 3: `readfile.c`

1. Создание программы

Все программы скомпилировали, последовательно выполнив пункты:

1. Создание программы

```
[root@wxcore wdobo]# su - guest
Last login: Sat Nov 13 22:46:21 MSK 2021 on pts/0
[guest@wxcore ~]$ vim simpleid.c
[guest@wxcore ~]$ gcc simpleid.c -o simpleid
[guest@wxcore ~]$ ls
dir1  simpleid  simpleid.c
[guest@wxcore ~]$ ./simpleid
uid=1001, gid=1001
[guest@wxcore ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest)
[guest@wxcore ~]$
```

Рис. 4: Пункты 3-5

1. Создание программы

```
[root@wxcore wdobo]# su - guest
Last login: Sat Nov 13 22:46:21 MSK 2021 on pts/0
[guest@wxcore ~]$ vim simpleid.c
[guest@wxcore ~]$ gcc simpleid.c -o simpleid
[guest@wxcore ~]$ ls
dir1  simpleid  simpleid.c
[guest@wxcore ~]$ ./simpleid
uid=1001, gid=1001
[guest@wxcore ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest)
[guest@wxcore ~]$ cp simpleid.c simpleid2.c
[guest@wxcore ~]$ vim simpleid2.c
[guest@wxcore ~]$ gcc simpleid2.c -o simpleid2
[guest@wxcore ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@wxcore ~]$
```

Рис. 5: Пункт 7

1. Создание программы

```
[guest@wxcore ~]$ exit
logout
[root@wxcore wdobo]# chown root:guest /home/guest/simpleid2
[root@wxcore wdobo]# chmod u+s /home/guest/simpleid2
[root@wxcore wdobo]# ls -l /home/guest/
total 52
drwxrwxrwx 2 guest guest  4096 Oct 30 17:30 dir1
-rwxrwxr-x 1 guest guest 17544 Nov 13 22:51 simpleid
-rwsrwxr-x 1 root  guest 17648 Nov 13 22:54 simpleid2
-rw-rw-r-- 1 guest guest   311 Nov 13 22:54 simpleid2.c
-rw-rw-r-- 1 guest guest   180 Nov 13 22:51 simpleid.c
[root@wxcore wdobo]# /home/guest
guest/  guest2/
[root@wxcore wdobo]# /home/guest/simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@wxcore wdobo]# su - guest
Last login: Sat Nov 13 22:51:44 MSK 2021 on pts/0
[guest@wxcore ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@wxcore ~]$
```

Рис. 6: Пункты 8-11

1. Создание программы

```
[guest@wxcore ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest)
[guest@wxcore ~]$ exit
logout
[root@wxcore wdobo]# chmod g+s /home/guest/simpleid2
[root@wxcore wdobo]# su - guest
Last login: Sat Nov 13 22:56:39 MSK 2021 on pts/0
[guest@wxcore ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@wxcore ~]$ ls -l
total 52
drwxrwxrwx 2 guest guest  4096 Oct 30 17:30 dir1
-rwxrwxr-x 1 guest guest 17544 Nov 13 22:51 simpleid
-rwsrwsr-x 1 root  guest 17648 Nov 13 22:54 simpleid2
-rw-rw-r-- 1 guest guest   311 Nov 13 22:54 simpleid2.c
-rw-rw-r-- 1 guest guest   180 Nov 13 22:51 simpleid.c
[guest@wxcore ~]$
```

Рис. 7: Пункт 12

1. Создание программы

```
[root@wxcore wdobo]# su - guest
Last login: Sat Nov 13 23:01:46 MSK 2021 on pts/0
[guest@wxcore ~]$ ls -l
total 76
drwxrwxrwx 2 guest guest  4096 Oct 30 17:30 dir1
-rwsrwxr-x 1 root  guest 17592 Nov 13 23:00 readfile
-rwx------ 1 root  guest   417 Nov 13 22:59 readfile.c
-rwxrwxr-x 1 guest guest 17544 Nov 13 22:51 simpleid
-rwsrwsr-x 1 root  guest 17648 Nov 13 22:54 simpleid2
-rw-rw-r-- 1 guest guest   311 Nov 13 22:54 simpleid2.c
-rw-rw-r-- 1 guest guest   180 Nov 13 22:51 simpleid.c
[guest@wxcore ~]$ cat readfile.c
cat: readfile.c: Permission denied
[guest@wxcore ~]$ exit
logout
[root@wxcore wdobo]# chown root:guest /home/guest/readfile
[root@wxcore wdobo]# chmod u+s /home/guest/readfile
[root@wxcore wdobo]# ls -l /home/guest/readfile
-rwsrwxr-x 1 root guest 17592 Nov 13 23:00 /home/guest/readfile
[root@wxcore wdobo]# su - guest
Last login: Sat Nov 13 23:04:20 MSK 2021 on pts/0
[guest@wxcore ~]$ ./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;
}
```

1. Создание программы

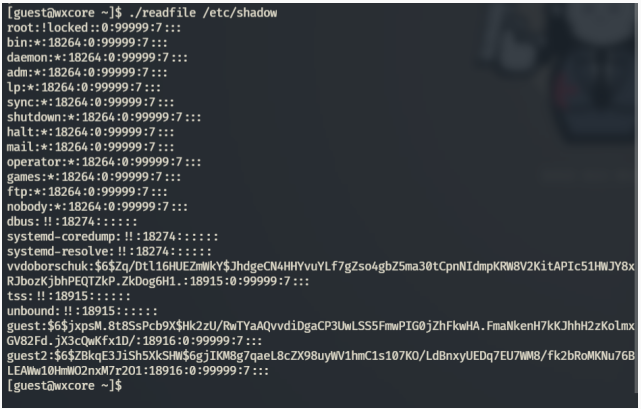


Рис. 9: Пункт 19

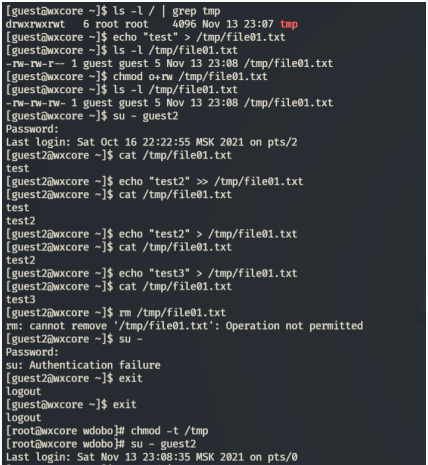
1. Создание программы

Команды `sudo` и `su` позволяют использовать права суперпользователя, или же использовать его учетную запись.

Как только мы установили SetUID-бит, программа readfile получила возможность читать файлы, принадлежащие суперпользователю.

2. Исследование Sticky-бита

Все действия по исследованию Sticky-бита удалось поместить в один скриншот:



2. Исследование Sticky-бита

В заключении, убрав Sticky-бит, мы смогли удалить файл чужого пользователя.

Заключение

Заключение

В результате выполнения работы мы изучили механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получили практические навыки работы в консоли с дополнительными атрибутами. Рассмотрели работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.