

Отчет по лабораторной работе 5

Дисциплина: Информационная безопасность

Хрусталеv Влад Николаевич

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	11

Список иллюстраций

Список таблиц

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

1. Создадим файл simpleid.c и введём в него программу

```
guest@vnkhrustalev:~ — vim 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;
}
```

```
[guest@vnkhrustalev ~]$ touch
[guest@vnkhrustalev ~]$ ls
dir1 Documents Pictures simpleid.c
[guest@vnkhrustalev ~]$ vim simpleid.c
```

2. Сохраним программу и сравним её работу с командой id. Как мы видим

вывод верный

```
[guest@vnkhrustalev ~]$ gcc simpleid.c -o simpleid
[guest@vnkhrustalev ~]$ ls
dir1 Documents Pictures simpleid simpleid.c
[guest@vnkhrustalev ~]$ ./simpleid
uid=1007, gid=100
[guest@vnkhrustalev ~]$ id
uid=1007(guest) gid=100(users) группы=100(users) контекст=unconfined_u:unconfine
d_r:unconfined_t:s0-s0:c0.c1023
[guest@vnkhrustalev ~]$
```

3. Создадим файл simpleid2.c и введём в него программу, далее скомпили-

```
guest@vnkhurstalev:~ — vim 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;
}
```

руем и запустим

```
guest@vnkhurstalev:~
[guest@vnkhurstalev ~]$ vim simpleid2.c
[guest@vnkhurstalev ~]$ gcc simpleid2.c -o simpleid2
[guest@vnkhurstalev ~]$ ls
dir1 Documents Pictures simpleid simpleid2 simpleid2.c simpleid.c
[guest@vnkhurstalev ~]$ ./simpleid2
e_uid=1007, e_gid=100
real_uid=1007, real_gid=100
[guest@vnkhurstalev ~]$
```

4. Изменим права доступа к файлу simpleid2, так и владельца на root.

После чего запустим программу и убедимся, что вывод схож с ко-

мандой id.

```
[vnkhurstalev@vnkhurstalev guest]$ sudo chown root:guest /home/guest/simpleid2
[vnkhurstalev@vnkhurstalev guest]$ sudo chmod u+s /home/guest/simpleid2
[vnkhurstalev@vnkhurstalev guest]$

[guest@vnkhurstalev ~]$ ls -l simpleid2
-rwsr-xr-x. 1 root guest 26048 anp 13 15:46 simpleid2
[guest@vnkhurstalev ~]$ ./simpleid2
e_uid=0, e_gid=100
real_uid=1007, real_gid=100
[guest@vnkhurstalev ~]$ id
uid=1007(guest) gid=100(users) группы=100(users) контекст=unconfined_u:unconfine
d_r:unconfined_t:s0-s0:c0.c1023
[guest@vnkhurstalev ~]$
```

5. Создадим файл `readfile.c` и введём в него программу. Далее изменим владельца на `root` и изменим права доступа так, чтобы доступ к файлу

```
guest@vnkhrustalev:~ — vim readfile.c
#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;
}
```

был только у `root`. :wq

```
[root@vnkhrustalev guest]# sudo chown root readfile.c
[root@vnkhrustalev guest]# sudo chmod 400 readfile.c
[root@vnkhrustalev guest]# ls
dir1 Pictures readfile.c simpleid2 simpleid.c
Documents readfile simpleid simpleid2.c
```

6. Попытаемся прочитать файл `readfile.c` от другого пользователя. У нас это не выйдет.(рис. ??)

```
[guest@vnkhrustalev ~]$ cat readfile.c
cat: readfile.c: Отказано в доступе
```

7. Добавим бит к файлу `readfile` и попытаемся из него прочитать файлы, к которым у нас нет доступа. Как видим, всё выполняется, т.к. мы устано-

```
[guest@vnkhrustalev ~]$ su
Пароль:
[root@vnkhrustalev guest]# sudo chown root readfile
[root@vnkhrustalev guest]# sudo chmod u+s readfile
[root@vnkhrustalev guest]#
```

вили SetUID-бит.


```
[root@vnkhurstalev guest]# ./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@vnkhurstalev ~]$ ./readfile /etc/shadow
root:$6$7JloaLwS18Hejtxo$E16/Ie3smJhU6nGZUp3X0
QGaDZcbEHqpG8qjshIPFXUfGLa3daR1:19679:0:99999:
bin:!:19469:0:99999:7:::
daemon:!:19469:0:99999:7:::
adm:!:19469:0:99999:7:::
lp:!:19469:0:99999:7:::
sync:!:19469:0:99999:7:::
```

8. Приступаем к следующей части работы. Создадим файл /tmp/file.01 от имени пользователя guest, изменим права, выдав доступ к чтению и записи

```
[root@vnkhurstalev tmp]# su guest
[guest@vnkhurstalev tmp]$ ls -l / | grep tmp
drwxrwxrwt. 15 root root 4096 anp 13 16:01 tmp
[guest@vnkhurstalev tmp]$ echo "test" > /tmp/file01.txt
[guest@vnkhurstalev tmp]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest users 5 anp 13 16:01 /tmp/file01.txt
[guest@vnkhurstalev tmp]$ chmod o+rw /tmp/file01.txt
[guest@vnkhurstalev tmp]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest users 5 anp 13 16:01 /tmp/file01.txt
[guest@vnkhurstalev tmp]$
```

группе OTHER.

9. Попробуем дописать этот файл, изменить полностью файл, удалить файл от пользователя guest2. У нас ничего этого не выйдет, т.к. guest2 в группе users, как и guest, а группе мы права не выдавали.

```
[guest2@vnkhurstalev tmp]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@vnkhurstalev tmp]$ echo "test3" > /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@vnkhurstalev tmp]$ echo "test3" >> /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@vnkhurstalev tmp]$ ls
file01.txt
```

```
[guest2@vnkhurstalev tmp]$ rm /tmp/file01.txt
rm: невозможно удалить '/tmp/file01.txt': Нет та
[guest2@vnkhurstalev tmp]$ rm /tmp/file01.txt
rm: невозможно удалить '/tmp/file01.txt': Нет та
[guest2@vnkhurstalev tmp]$ rm file01.txt
rm: удалить защищённый от записи обычный файл 'f
rm: невозможно удалить 'file01.txt': Операция не
[guest2@vnkhurstalev tmp]$
```

10. Изменим(удалим) Sticky бит папки /tmp и повторно попытаемся выполнить предыдущие манипуляции. Опять же ничего не выйдет, по той же причине. После всех наших манипуляций, вернём Sticky бит

```

[root@vnkhrustalev ~]# chmod -t /tmp
[root@vnkhrustalev ~]# exit
выход
[guest2@vnkhrustalev tmp]$ ls -l / | grep tmp
drwxrwxrwx. 18 root root 4096 апр 13 16:12 tmp
[guest2@vnkhrustalev tmp]$ cat /tmp/file01.txt
test3
[guest2@vnkhrustalev tmp]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@vnkhrustalev tmp]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@vnkhrustalev tmp]$ id
uid=1008(guest2) gid=100(users) группы=100(users),1005(guest) контекст=unconfine
d_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest2@vnkhrustalev tmp]$

```

для папки /tmp.

```

[guest2@vnkhrustalev tmp]$ su -
Пароль:
[root@vnkhrustalev ~]# chmod +t /tmp
[root@vnkhrustalev ~]# exit
выход

```

3 Выводы

Изучение механизмов изменения идентификаторов, применения SetUID- и Sticky-битов, а также получение практических навыков работы в консоли с дополнительными атрибутами позволяют глубже понять принципы безопасности и управления доступом в Unix-подобных системах. Рассмотрение работы механизма смены идентификатора процессов пользователей и влияние бита Sticky на запись и удаление файлов демонстрирует важность этих аспектов для обеспечения безопасности и контроля доступа в многопользовательских средах.