

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.

Задание

Выполнить задания лабораторной работы и проанализировать полученные результаты.

Теоретическое введение

Дискреционное управление доступом (англ. discretionary access control, DAC) — управление доступом субъектов к объектам на основе списков управления доступом или матрицы доступа. Также используются названия избирательное управление доступом, контролируемое управление доступом и разграничительное управление доступом.

Для каждой пары (субъект — объект) должно быть задано явное и недвусмысленное перечисление допустимых типов доступа, то есть тех типов доступа, которые являются санкционированными для данного субъекта (индивида или группы индивидов) к данному ресурсу (объекту).

Возможны несколько подходов к построению дискреционного управления доступом:

- Каждый объект системы имеет привязанного к нему субъекта, называемого владельцем. Именно владелец устанавливает права доступа к объекту.
- Система имеет одного выделенного субъекта — суперпользователя, который имеет право устанавливать права владения для всех остальных субъектов системы.
- Субъект с определённым правом доступа может передать это право любому другому субъекту.

Возможны и смешанные варианты построения, когда одновременно в системе присутствуют как владельцы, устанавливающие права доступа к своим объектам, так и суперпользователь, имеющий возможность изменения прав для любого объекта и/или изменения его владельца. Именно такой смешанный вариант реализован в большинстве операционных систем, например Unix.

Избирательное управление доступом является основной реализацией разграничительной политики доступа к ресурсам при обработке конфиденциальных сведений, согласно требованиям к системе защиты информации.

Выполнение лабораторной работы

Проверил, установлен ли у меня компилятор gcc и проследил, чтобы система защиты SELinux не мешала выполнению заданий работы.

Вошел в систему от имени пользователя guest и создал 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;
}
```

Скомпилировал программу:

```
[guest@vokhudickiyj ~]$ getenforce
Permissive
[guest@vokhudickiyj ~]$ touch simpleid.c
[guest@vokhudickiyj ~]$ vim simpleid.c
[guest@vokhudickiyj ~]$ gcc simpleid.c -o simpleid
```

Выполнил программу simpleid и системную программу id. Результаты совпали :

```
[guest@vokhudickiyj ~]$ ./simpleid
uid=1001, gid=1001
[guest@vokhudickiyj ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Усложнил 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 назвал simpleid2.c. Скомпилировал и запустил получившуюся программу:

```
[guest@vokhudickiyj ~]$ touch simpleid2.c
[guest@vokhudickiyj ~]$ vim simpleid2.c
[guest@vokhudickiyj ~]$ gcc simpleid2.c -o simpleid2
[guest@vokhudickiyj ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
```

От имени суперпользователя выполнил команды `chown root:guest /home/guest/simpleid2` и `chmod u+s /home/guest/simpleid2`:

```
[guest@vokhudickiyj ~]$ su
Password:
[root@vokhudickiyj guest]# chown root:guest /home/guest/simpleid2
[root@vokhudickiyj guest]# chmod u+s /home/guest/simpleid2
[root@vokhudickiyj guest]# ls -l simpleid2
-rwsrwxr-x. 1 root guest 26008 Oct  8 16:20 simpleid2
[root@vokhudickiyj guest]# su guest
[guest@vokhudickiyj ~]$
```

Команда `chown root:guest /home/guest/simpleid2` меняет владельца файла. Команда `chmod u+s /home/guest/simpleid2` меняет права доступа к файлу.

Запустил simpleid2 и id:

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

Команда `id` показывает действительные uid и gid.

Создал 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;
}
```

Откомпилировал её командой `gcc readfile.c -o readfile`:

```
[guest@vokhudickiyj ~]$ touch readfile.c
[guest@vokhudickiyj ~]$ vim readfile.c
[guest@vokhudickiyj ~]$ gcc readfile.c -o readfile
```

Сменил владельца у файла `readfile.c` и изменил права так, чтобы только суперпользователь мог прочитать его.

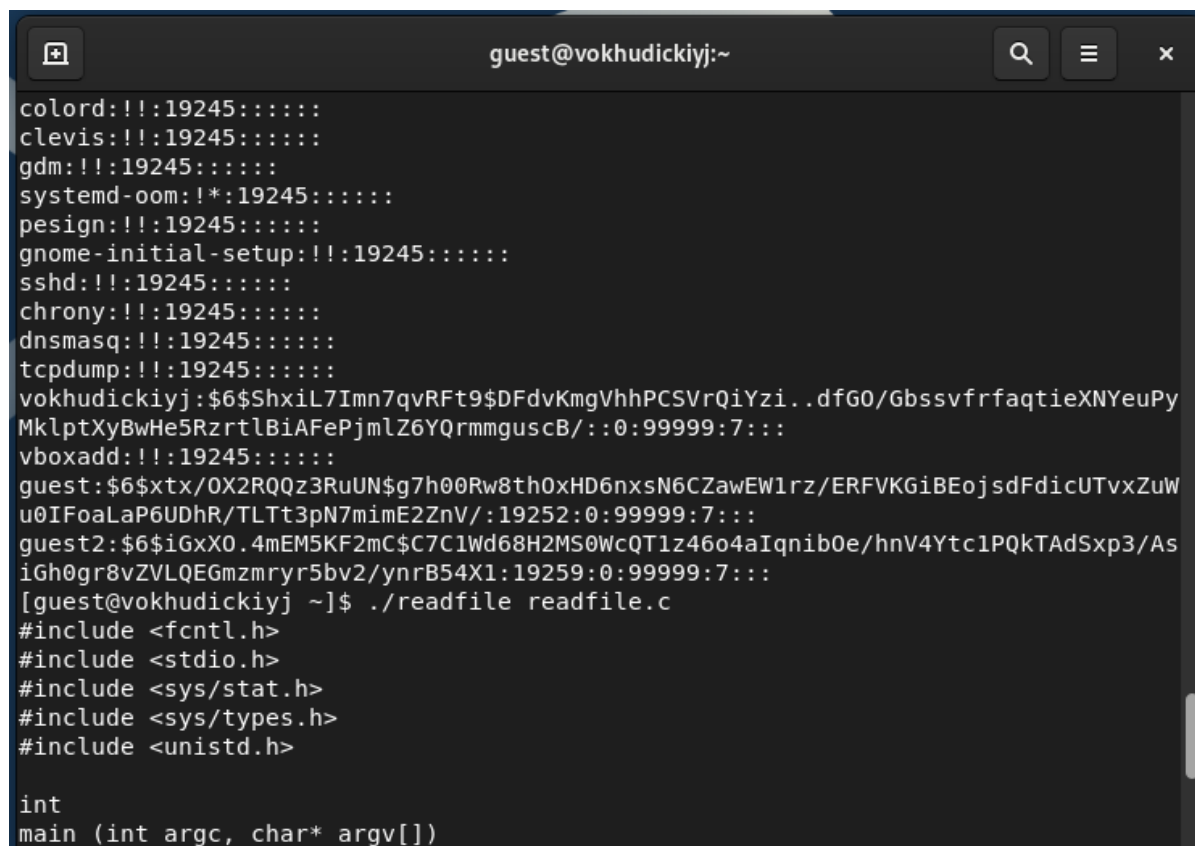
Проверил, что пользователь `guest` не может прочитать файл `readfile.c`:

```
[guest@vokhudickiyj ~]$ su
Password:
[root@vokhudickiyj guest]# chown root:guest readfile.c
[root@vokhudickiyj guest]# chmod 700 readfile.c
[root@vokhudickiyj guest]# su guest
[guest@vokhudickiyj ~]$ cat readfile.c
cat: readfile.c: Permission denied
```

Сменил у программы `readfile` владельца и установил SetUID-бит:

```
[guest@vokhudickiyj ~]$ su
Password:
[root@vokhudickiyj guest]# chown root:guest readfile
[root@vokhudickiyj guest]# chmod u+s readfile
```

Проверил, может ли программа `readfile` прочитать файлы `/etc/shadow` и `readfile.c`:



```
colord:!!:19245:~::~:
clevis:!!:19245:~::~:
gdm:!!:19245:~::~:
systemd-oom:!:*:19245:~::~:
pesign:!!:19245:~::~:
gnome-initial-setup:!!:19245:~::~:
sshd:!!:19245:~::~:
chrony:!!:19245:~::~:
dnsmasq:!!:19245:~::~:
tcpdump:!!:19245:~::~:
vokhudickiyj:$6$ShxiL7Imn7qvRft9$DFdvKmgVhhPCSvrQiYzi..dfG0/GbssvfrfaqtieXNYeuPy
MklptXyBwHe5RzrtlBiAFepJmLZ6YQrmmguscB/::0:99999:7:::
vboxadd:!!:19245:~::~:
guest:$6$xtx/0X2RQz3RuUN$g7h00Rw8th0xHD6nxsN6CZawEW1rz/ERFVKGiBEojsdFdicUTvxZuW
u0IFoaLaP6UDhR/TLTt3pN7mimE2ZnV/:19252:0:99999:7:::
guest2:$6$iGxX0.4mEM5KF2mC$C7C1Wd68H2MS0WcQT1z46o4aIqnib0e/hnV4Ytc1PQkTAdSxp3/As
iGh0gr8vZVLQEGmzmryr5bv2/ynrB54X1:19259:0:99999:7:::
[guest@vokhudickiyj ~]$ ./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[])
```

Так как у программы установлен SetUID-бит, то при её выполнении предоставляются права владельца файла (в данном случае `root`). Поэтому программа может прочитать файл с правами доступа только для `root`.

С помощью команды `ls -l | grep tmp` выяснил, установлен ли атрибут Sticky на директории `/tmp`:

```
[guest@vokhudickiyj ~]$ ls -l / | grep tmp
drwxrwxrwt. 16 root root 4096 Oct  8 16:58 tmp
```

От имени пользователя `guest` создал файл `file01.txt` в директории `/tmp` со словом `test` командой `echo "test" > /tmp/file01.txt`.

Просмотрел атрибуты у только что созданного файла и командой `chmod o+rw /tmp/file01.txt` разрешил чтение и запись для категории пользователей «все остальные».

```
[guest@vokhudickiyj ~]$ echo "test" > /tmp/file01.txt
[guest@vokhudickiyj ~]$ ls -l /tmp/file01.txt
-rw-rw-r--. 1 guest guest 5 Oct  8 17:02 /tmp/file01.txt
[guest@vokhudickiyj ~]$ chmod o+rw /tmp/file01.txt
[guest@vokhudickiyj ~]$ ls -l /tmp/file01.txt
-rw-rw-rw-. 1 guest guest 5 Oct  8 17:02 /tmp/file01.txt
```

От пользователя `guest2` (не являющегося владельцем) попробовал прочитать файл `/tmp/file01.txt`: `cat /tmp/file01.txt`.

От пользователя `guest2` попробовал дозаписать в файл `/tmp/file01.txt` слово `test2` командой `echo "test2" >> /tmp/file01.txt`. Мне удалось выполнить операцию.

Проверил содержимое файла командой `cat /tmp/file01.txt`

От пользователя `guest2` попробовал записать в файл `/tmp/file01.txt` слово `test3`, стерев при этом всю имеющуюся в файле информацию командой `echo "test3" > /tmp/file01.txt`. Мне удалось выполнить операцию.

Проверил содержимое файла командой `cat /tmp/file01.txt`.

От пользователя `guest2` попробовал удалить файл `/tmp/file01.txt` командой `rm /tmp/file01.txt`. Мне не удалось удалить файл:

```
[guest@vokhudickiyj ~]$ su guest2
Password:
[guest2@vokhudickiyj guest]$ cat /tmp/file01.txt
test
[guest2@vokhudickiyj guest]$ echo "test2" > /tmp/file01.txt
[guest2@vokhudickiyj guest]$ cat /tmp/file01.txt
test2
[guest2@vokhudickiyj guest]$ echo "test3" > /tmp/file01.txt
[guest2@vokhudickiyj guest]$ cat /tmp/file01.txt
test3
[guest2@vokhudickiyj guest]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': Operation not permitted
```

Повысил свои права до суперпользователя командой `su` и выполнил после этого команду `chmod -t /tmp`, снимающую атрибут `t` (Sticky-бит) с директории `/tmp`.

Повторил предыдущие шаги:

```
[guest2@vokhudickiyj guest]$ su
Password:
[root@vokhudickiyj guest]# chmod -t /tmp
[root@vokhudickiyj guest]# exit
exit
[guest2@vokhudickiyj guest]$ ls -l / | grep tmp
drwxrwxrwx. 16 root root 4096 Oct  8 17:10 tmp
[guest2@vokhudickiyj guest]$ cat /tmp/file01.txt
test3
[guest2@vokhudickiyj guest]$ echo "test2" > /tmp/file01.txt
[guest2@vokhudickiyj guest]$ cat /tmp/file01.txt
test2
[guest2@vokhudickiyj guest]$ echo "test3" > /tmp/file01.txt
[guest2@vokhudickiyj guest]$ cat /tmp/file01.txt
test3
[guest2@vokhudickiyj guest]$ rm /tmp/file01.txt
```

Мне удалось удалить файл от имени пользователя, не являющегося его владельцем, так как Sticky-bit позволяет защищать файлы от случайного удаления, когда несколько пользователей имеют права на запись в один и тот же каталог. Если у файла атрибут `t` стоит, значит пользователь может удалить файл, только если он является пользователем-владельцем файла или каталога, в котором содержится файл. Если же этот атрибут не установлен, то удалить файл могут все пользователи, которым позволено удалять файлы из каталога.

Повысил свои права до суперпользователя и вернул атрибут `t` на директорию `/tmp` с помощью команды `chmod +t /tmp`:

```
[guest2@vokhudickiyj guest]$ su
Password:
[root@vokhudickiyj guest]# chmod +t /tmp
[root@vokhudickiyj guest]# exit
exit
[guest2@vokhudickiyj guest]$ █
```


Выводы

В результате выполнения работы я изучил механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Рассмотрел работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

Список литературы

- [Кулябов Д. С., Королькова А. В., Геворкян М. Н. Лабораторная работа №5. Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов](#)