

**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЕТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5**

*дисциплина:* Основы информационной безопасности

Студент: Невзоров Дмитрий

**МОСКВА**

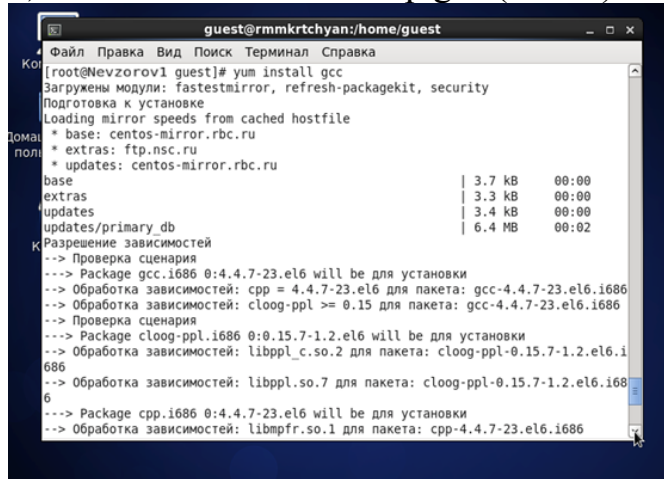
2021г.

## 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

## Ход работы:

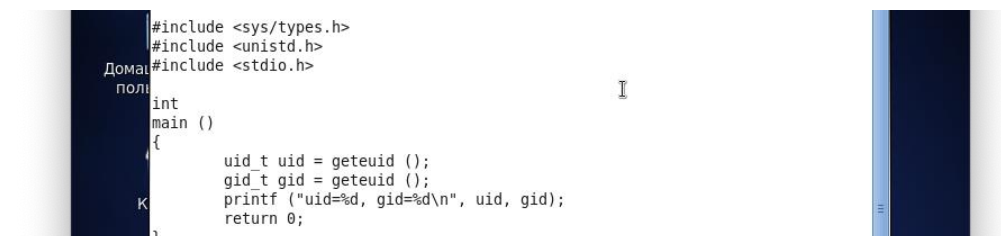
### 1) Установим компилятор gcc (Рис. 1)



```
guest@rmmkrtchyan:/home/guest
[root@Nevzorov1 guest]# yum install gcc
Загружены модули: fastestmirror, refresh-packagekit, security
Подготовка к установке
Loading mirror speeds from cached hostfile
 * base: centos-mirror.rbc.ru
 * extras: ftp.nsc.ru
 * updates: centos-mirror.rbc.ru
base                                     | 3.7 kB    00:00
extras                                 | 3.3 kB    00:00
updates                               | 3.4 kB    00:00
updates/primary_db                    | 6.4 MB    00:02
Разрешение зависимостей
--> Проверка сценария
--> Package gcc.i686 0:4.4.7-23.el6 will be для установки
--> Обработка зависимостей: crr = 4.4.7-23.el6 для пакета: gcc-4.4.7-23.el6.i686
--> Обработка зависимостей: cloog-ppl >= 0.15 для пакета: gcc-4.4.7-23.el6.i686
--> Проверка сценария
--> Package cloog-ppl.i686 0:0.15.7-1.2.el6 will be для установки
--> Обработка зависимостей: libppl_c.so.2 для пакета: cloog-ppl-0.15.7-1.2.el6.i686
--> Обработка зависимостей: libppl.so.7 для пакета: cloog-ppl-0.15.7-1.2.el6.i686
--> Package crr.i686 0:4.4.7-23.el6 will be для установки
--> Обработка зависимостей: libmpfr.so.1 для пакета: crr-4.4.7-23.el6.i686
```

Рис. 1

### 2) Создадим программу simpleid.c (Рис. 2)



```
#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;
}
```

Рис. 2

- 3) Скомпилируем программу (Рис. 3)  
gcc simpleid.c -o simpleid
- 4) Выполним програму simpleid (Рис. 3)  
./simpleid
- 5) Выполним системную программу id (Рис. 3)

```
[guest@Nevzorov1 ~]$ nano simpleid.c
[guest@Nevzorov1 ~]$ gcc simpleid.c -o simpleid
[guest@Nevzorov1 ~]$ ./simpleid
uid=501, gid=501
[guest@Nevzorov1 ~]$ id
uid=501(guest) gid=501(guest) группы=501(guest) контекст=unconfined_u:unconfined
r:unconfined t:s0-s0:c0.c1023
[guest@Nevzorov1 ~]$
```

Рис. 3

Результаты совпадают

- б) Усложним программу, добавив вывод действительных идентификаторов, сохранив как simpleid2.c (Рис. 4)

```
#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 = getuid ();
    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;
}
```

Рис. 4

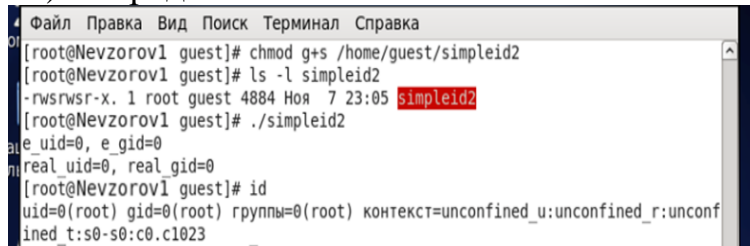
- 7) Скомпилируем и запустим simpleid2.c (Рис. 5)  
gcc simpleid2.c -o simpleid  
./simpleid2
- 8) От имени суперпользователя выполним команды (Рис. 5)  
chown root:guest /home/guest/simpleid2  
chmod u+s /home/guest/simpleid2  
С помощью этих команд файлу simpleid2 изменяем владельца и группу на root и guest соответственно, а также устанавливаем на файл SetUID-бит
- 9) Выполним проверку правильности установки новых атрибутов и смены владельца файла simpleid2 (Рис. 5)  
ls -l simpleid2
- 10) Запустим simpleid2 и id (Рис. 5)

./simpleid2

id

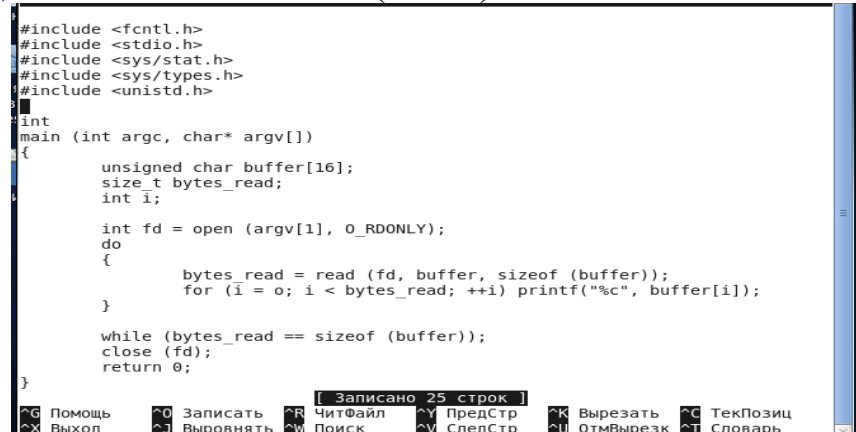
Результаты также одинаковы

## 11) Прделаем тоже самое относительно SetGID-бита



```
Файл Правка Вид Поиск Терминал Справка
[root@Nevzorov1 guest]# chmod g+s /home/guest/simpleid2
[root@Nevzorov1 guest]# ls -l simpleid2
-rwsrwsr-x. 1 root guest 4884 Ноя 7 23:05 simpleid2
[root@Nevzorov1 guest]# ./simpleid2
e uid=0, e gid=0
real uid=0, real gid=0
[root@Nevzorov1 guest]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

## 12) Создадим readfile.c (Рис. 7)



```
#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. 5

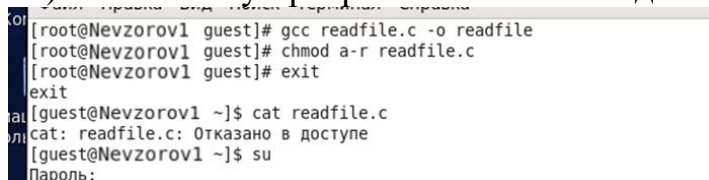
## 13) Откомпилируем ее (Рис. 8)

gcc readfile.c -o readfile

## 14) Сменим владельца у файла readfile.c и изменим права так, чтобы только root мог прочитать его (Рис. 8)

## 15) Проверим, что пользователь guest не может прочитать файл readfile.c (Рис. 8)

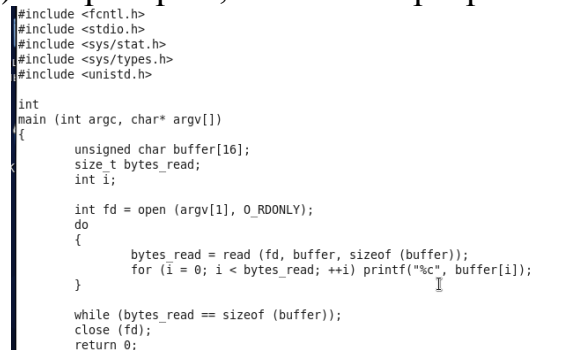
## 16) Сменим у программы readfile владельца и установим SetUID-бит (Рис. 8)



```
[root@Nevzorov1 guest]# gcc readfile.c -o readfile
[root@Nevzorov1 guest]# chmod a-r readfile.c
[root@Nevzorov1 guest]# exit
exit
[guest@Nevzorov1 ~]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@Nevzorov1 ~]$ su
Пароль:
```

Рис. 6

## 17) Проверим, может ли программа readfile прочитать файл readfile.c (Рис. 9)



```
#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. 7

## 18) Проверим, может ли программа readfile прочитать файл /etc/shadow (Рис. 9)

10)

```
V.RWr//RXKR.oAegLAg25dJdCBkZDA/:18152:0:99999:7:::  
bin:*:17246:0:99999:7:::  
daemon:*:17246:0:99999:7:::  
adm:*:17246:0:99999:7:::  
lp:*:17246:0:99999:7:::  
sync:*:17246:0:99999:7:::  
shutdown:*:17246:0:99999:7:::  
halt:*:17246:0:99999:7:::  
mail:*:17246:0:99999:7:::  
uucp:*:17246:0:99999:7:::  
operator:*:17246:0:99999:7:::  
games:*:17246:0:99999:7:::  
gopher:*:17246:0:99999:7:::  
ftp:*:17246:0:99999:7:::  
nobody:*:17246:0:99999:7:::  
dbus:!!:18152:.....:  
usbmuxd:!!:18152:.....:  
rpc:!!:18152:0:99999:7:::  
rtkit:!!:18152:.....:  
avahi-autoipd:!!:18152:.....:  
vcsa:!!:18152:.....:  
pulse:!!:18152:.....:
```

Рис. 8

19) Выясним, установлен ли атрибут Sticky на директории /tmp (Рис. 11)

```
ls -l / | grep tmp
```

20) От имени пользователя guest создайте файл file01.txt в директории /tmp со словом test (Рис. 11)

```
echo "test" > /tmp/file01.txt
```

21) Просмотрим атрибуты у только что созданного файла и разрешим чтение и запись для категории пользователей «все остальные» (Рис. 11)

```
ls -l /tmp/file01.txt
```

```
chmod o+rw /tmp/file01.txt
```

```
ls -l /tmp/file01.txt
```

```
[guest@Nevzorov1 ~]$ ls -l / | grep tmp  
drwxrwxrwt. 34 root root 4096 Ноя 8 00:40 tmp  
[guest@Nevzorov1 ~]$ echo "test" > /tmp/file01.txt  
[guest@Nevzorov1 ~]$ ls -l /tmp/file01.txt  
-rw-rw-r--. 1 guest guest 5 Ноя 8 00:40 /tmp/file01.txt  
[guest@Nevzorov1 ~]$ chmod o+rw /tmp/file01.txt
```

Рис. 9

22) От пользователя guest2 (не являющегося владельцем) попробуйте прочитать файл /tmp/file01.txt

```
cat /tmp/file01.txt
```

23) От пользователей guest2 попробуем дозаписать в файл /tmp/file01.txt слово test2

```
echo "test2" > /tmp/file01.txt
```

Дозаписать не получилось

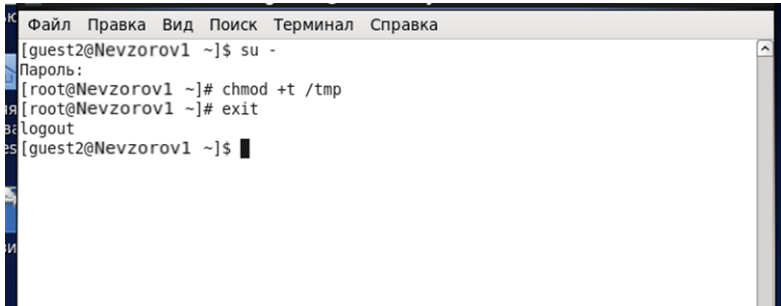
24) Проверим содержимое файла

```
cat /tmp/file01.txt
```

25) От пользователей guest2 попробуем записать в файл /tmp/file01.txt слово test3, стерев всю информацию в файле

```
echo "test3" > /tmp/file01.txt
```

- Перезаписать информацию получилось
- 26) Проверим содержимое файла  
`cat /tmp/file01.txt`
- 27) От пользователя `guest2` попробуйте удалить файл `/tmp/file01.txt`  
`rm /tmp/file01.txt`  
Файл не удалился
- 28) Повысим свои права до суперпользователя  
`su -`  
И выполним после этого команду снимающую атрибут Sticky-бита с директории `/tmp`  
`chmod -t /tmp`
- 29) Покинем режим суперпользователя  
`exit`
- 30) От пользователя `guest2` проверим, что атрибут `t` у директории `/tmp` нет  
`ls -l / | grep tmp`
- 31) Повторим предыдущие шаги  
Файл удалось удалить от имени пользователя, не являющегося его владельцем
- 32) Повысим свои права до суперпользователя и вернем атрибут `t` на директорию `/tmp`  
`su -`  
`chmod +t /tmp`  
`exit`



```
Кс  Файл  Правка  Вид  Поиск  Терминал  Справка
[guest2@Nevzorov1 ~]$ su -
Пароль:
[root@Nevzorov1 ~]# chmod -t /tmp
[root@Nevzorov1 ~]# exit
logout
[guest2@Nevzorov1 ~]$
```

Рис. 10

## Вывод:

Я изучил механизмы изменения идентификаторов, применения SetUID- и Sticky-битов, получил практические навыки работы в консоли с дополнительными атрибутами, а также рассмотрели работы механизма смены идентификатора процессов пользователей и влияние бита Sticky на запись и удаление файлов