

# Отчёт по лабораторной работе №5

## Информационная безопасность

### Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Выполнила: Сингх Ааруши ,  
НКАбд-02-23, 132215095

#### Содержание

Цель работы .....	1
Теоретическое введение .....	1
Выполнение лабораторной работы .....	3
5.2.1. Подготовка лабораторного стенда.....	3
5.3.1 Создание программы .....	3
5.3.2. Исследование Sticky-бита.....	9
Вывод.....	11
Список литературы. Библиография .....	11

#### 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.

#### Теоретическое введение

##### 1. Дополнительные атрибуты файлов Linux

В Linux существует три основных вида прав — право на чтение (read), запись (write) и выполнение (execute), а также три категории пользователей, к которым они могут применяться — владелец файла (user), группа владельца (group) и все остальные (others). Но, кроме прав чтения, выполнения и записи, есть еще три дополнительных атрибута. [1]

- **Sticky bit**

Используется в основном для каталогов, чтобы защитить в них файлы. В такой каталог может писать любой пользователь. Но, из такой директории пользователь может удалить только те файлы, владельцем которых он является. Примером может служить директория /tmp, в которой запись открыта для всех пользователей, но нежелательно удаление чужих файлов.

- **SUID (Set User ID)**

Атрибут исполняемого файла, позволяющий запустить его с правами владельца. В Linux приложение запускается с правами пользователя, запустившего указанное приложение. Это обеспечивает дополнительную безопасность т.к. процесс с правами пользователя не сможет получить доступ к важным системным файлам, которые принадлежат пользователю root.

- **SGID (Set Group ID)**

Аналогичен suid, но относится к группе. Если установить sgid для каталога, то все файлы созданные в нем, при запуске будут принимать идентификатор группы каталога, а не группы владельца, который создал файл в этом каталоге.

- **Обозначение атрибутов sticky, suid, sgid**

Специальные права используются довольно редко, поэтому при выводе программы ls -l символ, обозначающий указанные атрибуты, закрывает символ стандартных прав доступа.

Пример:

```
rwsrwsrwt
```

*где первая s — это suid, вторая s — это sgid, а последняя t — это sticky bit*

В приведенном примере не понятно, rwt — это gw- или gwx? Определить это просто. Если t маленькое, значит x установлен. Если T большое, значит x не установлен. То же самое правило распространяется и на s.

В числовом эквиваленте данные атрибуты определяются первым символом при четырехзначном обозначении (который часто опускается при назначении прав), например в правах 1777 — символ 1 обозначает sticky bit. Остальные атрибуты имеют следующие числовое соответствие:

- 1 — установлен sticky bit
- 2 — установлен sgid
- 4 — установлен suid

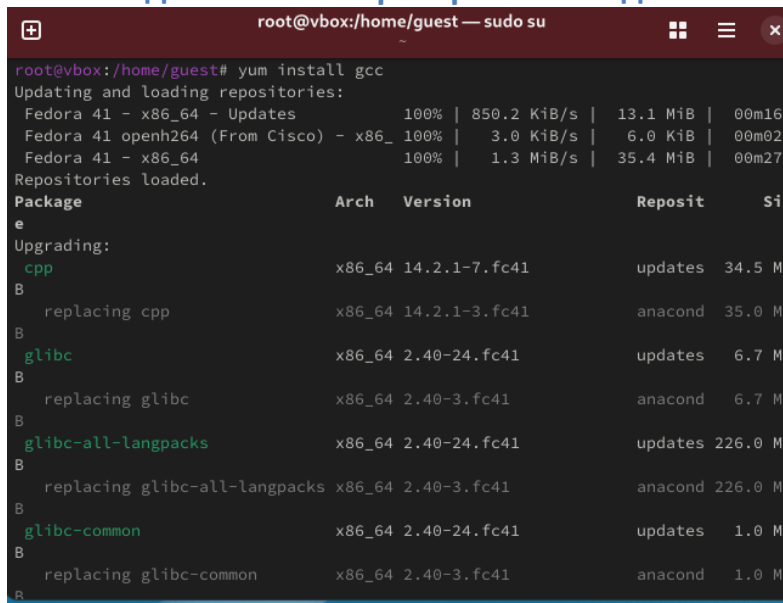
## **2. Компилятор GCC**

GCC - это свободно доступный оптимизирующий компилятор для языков C, C++. Собственно программа gcc это некоторая надстройка над группой компиляторов, которая способна анализировать имена файлов, передаваемые ей в качестве аргументов, и определять, какие действия необходимо выполнить. Файлы с

расширением .cc или .C рассматриваются, как файлы на языке C++, файлы с расширением .c как программы на языке C, а файлы с расширением .o считаются объектными. [2]

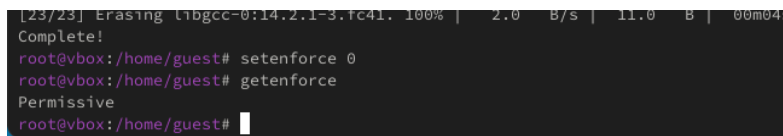
## Выполнение лабораторной работы

### 5.2.1. Подготовка лабораторного стенда



```
root@vbox:/home/guest — sudo su
root@vbox:/home/guest# yum install gcc
Updating and loading repositories:
Fedora 41 - x86_64 - Updates          100% | 850.2 KiB/s | 13.1 MiB | 00m16s
Fedora 41 openh264 (From Cisco) - x86_ 100% | 3.0 KiB/s | 6.0 KiB | 00m02s
Fedora 41 - x86_64                   100% | 1.3 MiB/s | 35.4 MiB | 00m27s
Repositories loaded.
Package Arch Version Repository Size
e
Upgrading:
  cpp x86_64 14.2.1-7.fc41 updates 34.5 MiB
B replacing cpp x86_64 14.2.1-3.fc41 anacond 35.0 MiB
B
  glibc x86_64 2.40-24.fc41 updates 6.7 MiB
B replacing glibc x86_64 2.40-3.fc41 anacond 6.7 MiB
B
  glibc-all-langpacks x86_64 2.40-24.fc41 updates 226.0 MiB
B replacing glibc-all-langpacks x86_64 2.40-3.fc41 anacond 226.0 MiB
B
  glibc-common x86_64 2.40-24.fc41 updates 1.0 MiB
B replacing glibc-common x86_64 2.40-3.fc41 anacond 1.0 MiB
B
```

(рис. 1. Установка gss)



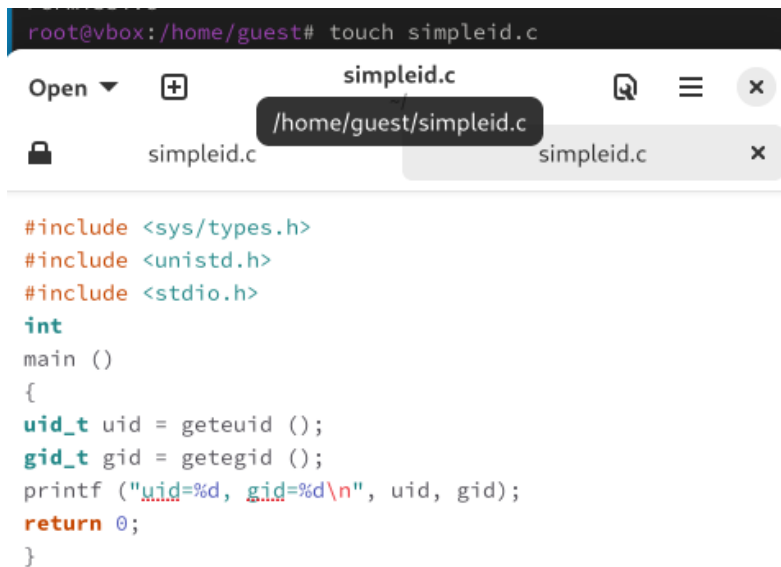
```
[23/23] Erasing libgcc-0:14.2.1-3.fc41. 100% | 2.0 B/s | 11.0 B | 00m04s
Complete!
root@vbox:/home/guest# setenforce 0
root@vbox:/home/guest# getenforce
Permissive
root@vbox:/home/guest#
```

(рис. 1. Установка gss)

### 5.3.1 Создание программы

1. Войдите в систему от имени пользователя guest.
2. Создайте программу simpleid.c.

```
root@vbox:/home/guest# touch 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;
}
```

(рис. 2. simpleid.c)

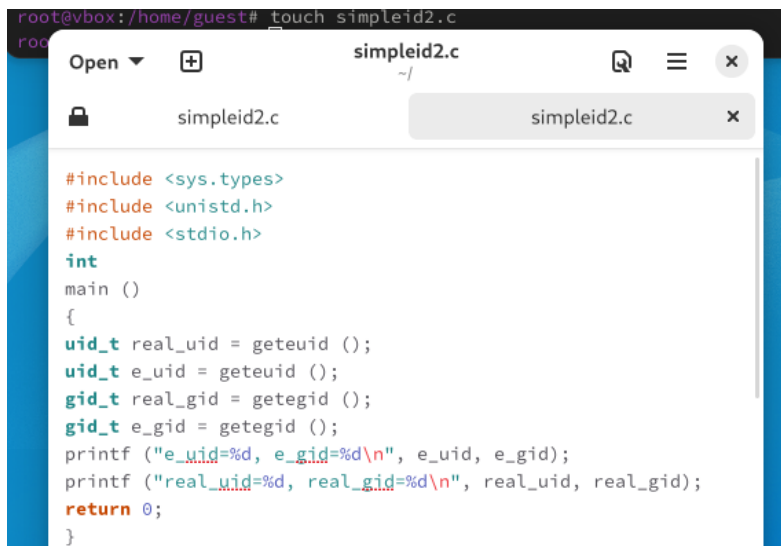
3. Скомпилируйте программу и убедитесь, что файл программы создан: `gcc simpleid.c -o simpleid`
4. Выполните программу simpleid: `./simpleid`
5. Выполните системную программу `id`: `id` и сравните полученный вами результат с данными предыдущего пункта задания.

```
root@vbox:/home/guest# gcc simpleid.c -o simpleid
root@vbox:/home/guest# ./simpleid
uid=0, gid=0
root@vbox:/home/guest# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfi
ned_t:s0-s0:c0.c1023
root@vbox:/home/guest#
```

(рис. 3. 3-5 пункты задания лабораторной)

6. Усложните программу, добавив вывод действительных идентификаторов.

```
root@vbox:/home/guest# touch simpleid2.c
root@vbox:/home/guest#
```



```
#include <sys/types>
#include <unistd.h>
#include <stdio.h>
int
main ()
{
    uid_t real_uid = getuid ();
    uid_t e_uid = seteuid ();
    gid_t real_gid = getgid ();
    gid_t e_gid = setegid ();
    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. simpleid2.c)

7. Скомпилируйте и запустите simpleid2.c: `gcc simpleid2.c -o simpleid2 ./simpleid2`

```
root@vbox:/home/guest# gcc simpleid2.c -o simpleid2
root@vbox:/home/guest# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
root@vbox:/home/guest#
```

(рис. 5. 7 пункт задания лабораторной)

8. От имени суперпользователя выполните команды: `chown root:guest /home/guest/simpleid2` `chmod u+s /home/guest/simpleid2`
9. Используйте `sudo` или повысьте временно свои права с помощью `su`. Поясните, что делают эти команды.

От имени суперпользователя выполнила команды “`sudo chown root:guest /home/guest/simpleid2`” и “`sudo chmod u+s /home/guest/simpleid2`”, затем выполнила проверку правильности установки новых атрибутов и смены владельца файла `simpleid2` командой “`sudo ls -l /home/guest/simpleid2`” (рис. 3.9). Этими командами была произведена смена пользователя файла на `root` и установлен SetUID-бит.

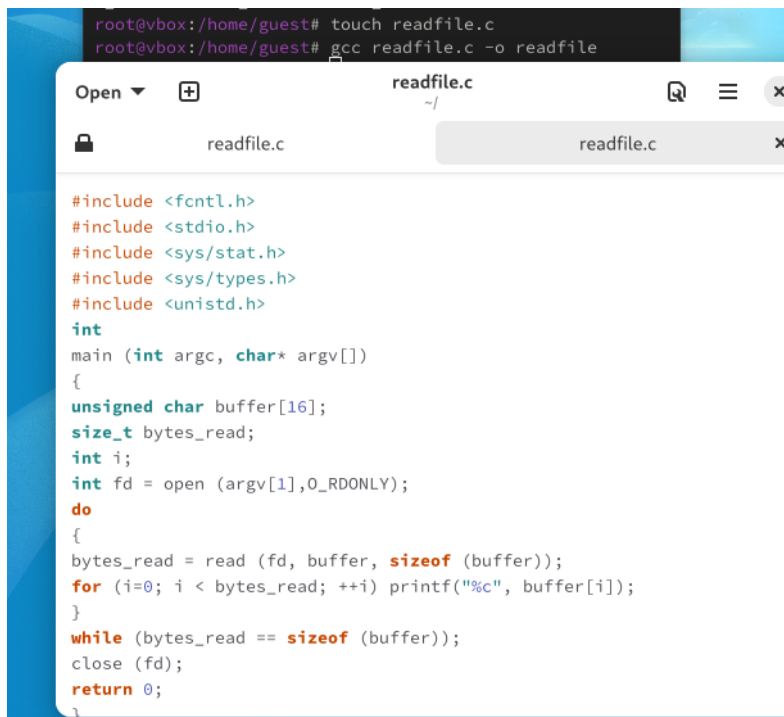
10. Выполните проверку правильности установки новых атрибутов и смены владельца файла `simpleid2`: `ls -l simpleid2`
11. Запустите `simpleid2` и `id`: `./simpleid2 id` Сравните результаты.
12. Прделайте тоже самое относительно SetGID-бита.

```
root@vbox:/home/guest# chown root:guest simpleid2
root@vbox:/home/guest# chmod u+s simpleid2
root@vbox:/home/guest# ls -l simpleid2
-rwsr-xr-x. 1 root guest 16728 Apr 26 20:56 simpleid2
root@vbox:/home/guest# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
root@vbox:/home/guest# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfi
ned_t:s0-s0:c0.c1023
root@vbox:/home/guest# chown root:guesr simpleid2
chown: invalid group: 'root:guesr'
root@vbox:/home/guest# chown root:guest simpleid2
root@vbox:/home/guest# chmod g+s simpleid2
root@vbox:/home/guest# ls -l simpleid2
-rwxr-sr-x. 1 root guest 16728 Apr 26 20:56 simpleid2
root@vbox:/home/guest# ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=1001
root@vbox:/home/guest# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfi
ned_t:s0-s0:c0.c1023
root@vbox:/home/guest#
```

(рис. 6. 8-12 пункты задания лабораторной)

13. Создайте программу readfile.c

14. Откомпилируйте её. gcc readfile.c -o readfile



```
root@vbox:/home/guest# touch readfile.c
root@vbox:/home/guest# gcc readfile.c -o readfile

Open ▾ + readfile.c ~/
readfile.c readfile.c x

#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. readfile.c)

15. Смените владельца у файла readfile.c (или любого другого текстового файла в системе) и измените права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог.

```
root@vbox:/home/guest# su
root@vbox:/home/guest# chown root:guest readfile
root@vbox:/home/guest# chmod 700 readfile
root@vbox:/home/guest# chown root:guest readfile
root@vbox:/home/guest# chmod -r readfile.c
root@vbox:/home/guest# chmod u+c readfile
chmod: invalid mode: 'u+c'
Try 'chmod --help' for more information.
root@vbox:/home/guest# chmod u+s readfile
root@vbox:/home/guest#
```

(рис. 8. chmod)

16. Проверьте, что пользователь guest не может прочитать файл readfile.c.
17. Смените у программы readfile владельца и установите SetU'D-бит.
18. Проверьте, может ли программа readfile прочитать файл readfile.c?
19. Проверьте, может ли программа readfile прочитать файл /etc/shadow?  
Отразите полученный результат и ваши объяснения в отчёте.

```
guest@vbox:~$ cat readfile.c
cat: readfile.c: Permission denied
guest@vbox:~$ ./readfile readfile.c
bash: ./readfile: Permission denied
guest@vbox:~$ ./readfile/etc/shadow
bash: ./readfile/etc/shadow: Not a directory
guest@vbox:~$
```

(рис. 9. 16-19 пункты Guest)

От имени суперпользователя все команды удастся выполнить.

```
guest@vbox:~$ sudo su
[sudo] password for guest:
root@vbox:/home/guest# 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;
}
root@vbox:/home/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
```

*(рис. 10. 16-18 пункты суперпользователь)*



```
root@vbox:/home/guest# ./readfile /etc/shadow
root::!:0:99999:7:::
bin:*:19925:0:99999:7:::
daemon:*:19925:0:99999:7:::
adm:*:19925:0:99999:7:::
lp:*:19925:0:99999:7:::
sync:*:19925:0:99999:7:::
shutdown:*:19925:0:99999:7:::
halt:*:19925:0:99999:7:::
mail:*:19925:0:99999:7:::
operator:*:19925:0:99999:7:::
games:*:19925:0:99999:7:::
ftp:*:19925:0:99999:7:::
nobody:*:19925:0:99999:7:::
dbus:!:20020:::
apache:!:20020:::
tss:!:20020:::
avahi:!:20020:::
geoclue:!:20020:::
usbmuxd:!:20020:::
systemd-oom:!*:20020:::
qemu:!:20020:::
polkitd:!:20020:::
rtkit:!:20020:::
chrony:!:20020:::
dnsmasq:!:20020:::
gluster:!:20020:::
rpc:!:20020:0:99999:7:::
pipewire:!:20020:::
unbound:!:20020:::
nm-openconnect:!:20020:::
rpcuser:!:20020:::
wsdd:!:20020:::
sssd:!:20020:::
openvpn:!:20020:::
nm-openvpn:!:20020:::
```

(рис. 11. 19 пункт суперпользователь)

### 5.3.2. Исследование Sticky-бита

1. Выясните, установлен ли атрибут Sticky на директории /tmp, для чего выполните команду `ls -l / | grep tmp`
2. От имени пользователя guest создайте файл file01.txt в директории /tmp со словом test: `echo "test" > /tmp/file01.txt`
3. Просмотрите атрибуты у только что созданного файла и разрешите чтение и запись для категории пользователей «все остальные»: `ls -l /tmp/file01.txt`  
`chmod o+rw /tmp/file01.txt`  
`ls -l /tmp/file01.txt`

```
root@vbox:/home/guest# ls -l / | grep tmp
drwxrwxrwt. 20 root root 500 Apr 26 21:14 tmp
root@vbox:/home/guest# echo "test" > /tmp/file01.txt
root@vbox:/home/guest# ls -l /tmp/file01.txt
-rw-r--r--. 1 root root 5 Apr 26 21:16 /tmp/file01.txt
root@vbox:/home/guest# chmod o+rw /tmp/file01.txt
root@vbox:/home/guest# ls -l /tmp/file01.txt
-rw-r--rw-. 1 root root 5 Apr 26 21:16 /tmp/file01.txt
root@vbox:/home/guest#
```

(рис. 12. 1-3 пункты)

4. От пользователя guest2 (не являющегося владельцем) попробуйте прочитать файл /tmp/file01.txt: cat /tmp/file01.txt
5. От пользователя guest2 попробуйте дозаписать в файл /tmp/file01.txt слово test2 командой echo "test2" > /tmp/file01.txt

Удалось ли вам выполнить операцию? Нет.

6. Проверьте содержимое файла командой cat /tmp/file01.txt
7. От пользователя guest2 попробуйте записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой echo "test3" > /tmp/file01.txt

Удалось ли вам выполнить операцию? Нет.

8. Проверьте содержимое файла командой cat /tmp/file01.txt
9. От пользователя guest2 попробуйте удалить файл /tmp/file01.txt командой rm /tmp/file01.txt

Удалось ли вам удалить файл? Нет.

10. Повысьте свои права до суперпользователя следующей командой su и выполните после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp: chmod -t /tmp
11. Покиньте режим суперпользователя командой exit
12. От пользователя guest2 проверьте, что атрибута t у директории /tmp нет: ls -l / | grep tmp

```
guest2@vbox:~$ sudo su
[sudo] password for guest2:
root@vbox:/home/guest2# chmod -t /tmp
root@vbox:/home/guest2# exit
exit
guest2@vbox:~$ ls -l / | grep tmp
drwxrwxrwx. 22 root root 640 Apr 26 21:28 tmp
guest2@vbox:~$
```

(рис. 13. 4-12 пункты)

13. Повторите предыдущие шаги. Какие наблюдаются изменения?

При повторении всё получилось.

14. Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем? Удалось.

15. Повысьте свои права до суперпользователя и верните атрибут t на директорию /tmp: su chmod +t /tmp exit

```
guest2@vbox:~$ sudo su
root@vbox:/home/guest2# chmod +t /tmp
root@vbox:/home/guest2# exit
exit
guest2@vbox:~$
```

(рис. 14. Возвращение атрибута)

## Вывод

Были изучены механизмы изменения идентификаторов и применения SetUID- и Sticky-битов. Получены практические навыки работы в консоли с дополнительными атрибутами. Были рассмотрены работа механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

## Список литературы. Библиография

[0] Методические материалы курса

[1] Дополнительные атрибуты: <https://tokmakov.msk.ru/blog/item/141>

[2] Компилятор GSS: <http://parallel.imm.uran.ru/freesoft/make/instrum.html>