

# Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

---

Савченко Елизавета НБИ-01-20

7 октября, 2023, Москва, Россия

Российский Университет Дружбы Народов

# Цели и задачи

---

- SUID - разрешение на установку идентификатора пользователя. Это бит разрешения, который позволяет пользователю запускать исполняемый файл с правами владельца этого файла.
- SGID - разрешение на установку идентификатора группы. Принцип работы очень похож на SUID с отличием, что файл будет запускаться пользователем от имени группы, которая владеет файлом.

## Цель лабораторной работы

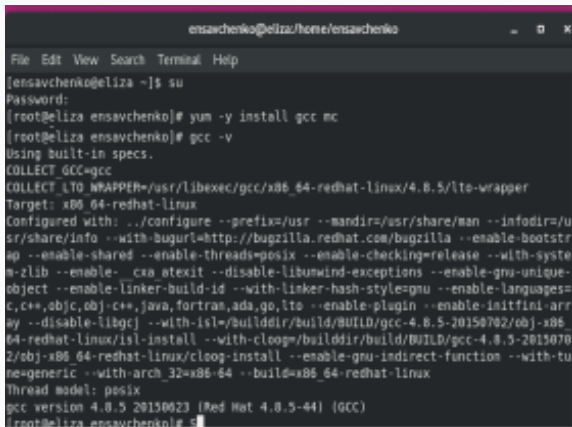
Изучение механизмов изменения идентификаторов, применения SetUID и Sticky-битов. Получение практических навыков работы в консоли с дополнительными атрибутами. Рассмотрение работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

# **Выполнение лабораторной работы**

---



Для выполнения части заданий требуются средства разработки приложений. Проверили наличие установленного компилятора gcc командой `gcc -v`: компилятор обнаружен.



```
ensavchenko@eliza:~/home/ensavchenko
File Edit View Search Terminal Help
[ensavchenko@eliza ~]$ su
Password:
[root@eliza ensavchenko]# yum -y install gcc nc
[root@eliza ensavchenko]# gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/4.8.5/lto-wrapper
Target: x86_64-redhat-linux
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --with-bugurl=http://bugzilla.redhat.com/bugzilla --enable-bootstrap --enable-shared --enable-threads=posix --enable-checking=release --with-system-zlib --enable-__cxa_atexit --disable-libunwind-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-linker-hash-style=gnu --enable-languages=c,c++,objc,obj-c++,java,fortran,ada,go,lto --enable-plugin --enable-initfini-array --disable-libgomp --with-isl=/build/builddir/build/BUILD/gcc-4.8.5-20150702/obj-x86_64-redhat-linux/isl-install --with-cloog=/build/builddir/build/BUILD/gcc-4.8.5-20150702/obj-x86_64-redhat-linux/cloog-install --enable-gnu-indirect-function --with-tune=generic --with-arch_32=x86_64 --build=x86_64-redhat-linux
Thread model: posix
gcc version 4.8.5 20150623 (Red Hat 4.8.5-44) (GCC)
[root@eliza ensavchenko]#
```

Figure 1: установка компилятора

**Чтобы система защиты SELinux не мешала выполнению заданий работы, отключили систему запретов до очередной перезагрузки системы командой `setenforce 0`:**



## Команда `getenforce` вывела `Permissive`:

```
gcc version 4.8.5 20150623 (Red Hat 4.8.5-44) (GCC)
[root@eliza ensavchenko]# setenforce 0
[root@eliza ensavchenko]# getenforce
Permissive
```

**Figure 2:** подготовка к работе

## Теперь проверим корректность установки компилятора, как показано на скриншоте номер 3.

```
[root@eliza ensavchenko]# gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/4.8.5/lto-wrapper
Target: x86_64-redhat-linux
Configured with: ../configure --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --with-bugurl=http://bugzilla.redhat.com/bugzilla --enable-bootstrap --enable-shared --enable-threads=posix --enable-checking=release --with-system-zlib --enable-_cxa_atexit --disable-libunwind-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-linker-hash-style=gnu --enable-languages=c,c++,objc,obj-c++,java,fortran,ada,go,lto --enable-plugin --enable-initfini-array --disable-libgcj --with-isl=/builddir/build/BUILD/gcc-4.8.5-20150702/obj-x86_64-redhat-linux/isl-install --with-cloog=/builddir/build/BUILD/gcc-4.8.5-20150702/obj-x86_64-redhat-linux/cloog-install --enable-gnu-indirect-function --with-tune=generic --with-arch_32=x86_64 --build=x86_64-redhat-linux
Thread model: posix
gcc version 4.8.5 20150623 (Red Hat 4.8.5-44) (GCC)
[root@eliza ensavchenko]# getenforce
Permissive
```

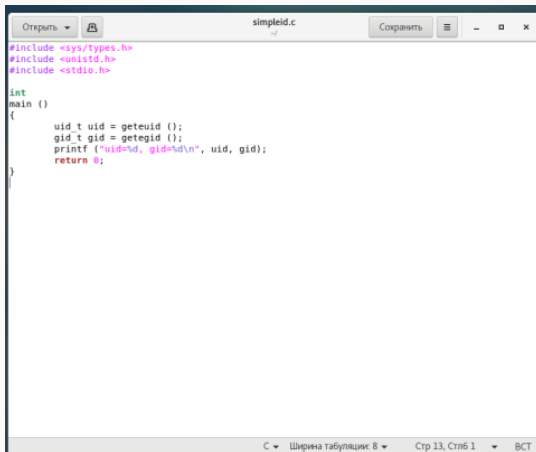
**Figure 3:** проверка установки компилятора

1. Вошли в систему от имени пользователя guest.

```
[guest@eliza ~]$ mkdir laba5
mkdir: cannot create directory 'laba5': File exists
[guest@eliza ~]$ touch simpleid.c
[guest@eliza ~]$ cat simpleid.c
#include <stdio.h>
```

**Figure 4:** создание и редактирование файла simpleid.c

# Написали 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;
}
```

**Figure 5:** написание программы

Скомпилировали программу и убедились, что файл программы создан: `gcc simpleid.c -o simpleid`

Выполнили программу `simpleid` командой `./simpleid`

## Выполнили системную программу `id` с помощью команды `id`. `uid` и `gid` совпадает в обеих программах

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

**Figure 6:** результат программы `simpleid`

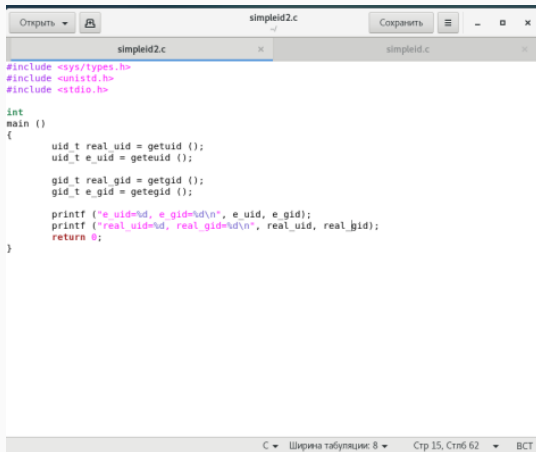
## Усложнили программу, добавив вывод действительных идентификаторов.

```
[guest@eliza ~]$ touch simpleid2.c  
[guest@eliza ~]$ gedit simpleid2.c
```

**Figure 7:** Создание и редактирование файла simpleid2.c



# Написали следующую программу в файле 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;
}
```

**Figure 8:** написание программы

## Скомпилировали и запустили simpleid2.c: gcc

```
[guest@eliza ~]$ gcc simpleid2.c -o simpleid2  
[guest@eliza ~]$ ./simpleid2  
e_uid=1001, e_gid=1001  
real uid=1001, real_gid=1001
```

**Figure 9:** результат программы simpleid2 от guest

На скриншоте 10 видно следующее: теперь от имени суперпользователя выполнили команды, которые меняют владельца файла. Использовали `su` для повышения

прав до суперпользователя. Выполнили проверку правильности установки новых атрибутов и смены владельца файла `simpleid2`. Также запустили `simpleid2` и `id`.

```
[guest@eliza ~]$ su
Password:
[root@eliza guest]# chown root:guest /home/guest/simpleid2
[root@eliza guest]# chmod u+s /home/guest/simpleid2
[root@eliza guest]# ls -l simpleid2
-rwsrwxr-x. 1 root guest 8616 Oct  7 22:39 simpleid2
[root@eliza guest]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@eliza guest]#
```

**Figure 10:** результат программы `simpleid2` от root

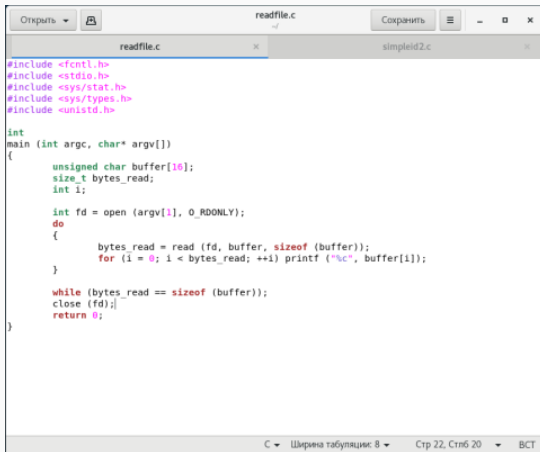
Приходим к выводу, что результат выполнения программ при заходе от пользователя `guest` и через `root` отличается.

# Создание и редактирование файла

```
[guest@eliza ~]$ touch readfile.c  
[guest@eliza ~]$ gedit readfile.c  
[guest@eliza ~]$
```

**Figure 11:** создание и редактирование файла readfile.c

# Написали 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[10];
    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;
}
```

Figure 12: программа readfile

## Откомпилировали программу. Затем сменили владельца у файла `readfile.c` и

изменили права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог.

```
[root@eliza guest]# chown root:root readfile
[root@eliza guest]# chown o-r readfile.c
chown: invalid user: 'o-r'
[root@eliza guest]# chmod o-r readfile.c
[root@eliza guest]# chmod g-rw readfile.c
[root@eliza guest]# chmod u+s readfile
[root@eliza guest]# exit
exit
```

**Figure 13:** смена прав на файле `readfile`

# Проверили, что пользователь guest не может прочитать файл readfile.c. Но

увидели, что файл читается.

```
[guest@eliza ~]$ 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;
}
[guest@eliza ~]$
```

**Figure 14:** проверка файла на чтение

**Ищем ошибку. Оказалось, что была допущена ошибка в установке прав. Переделали команду с установкой прав и повторили проверку, на этот раз в доступе**

отказано, как и должно быть. А команда `./readfile` сработала потому что она выполняет чтение от имени суперпользователя, а не `guest`.





## От имени пользователя `guest` создали файл `file01.txt` в директории `/tmp` со словом

`test`. Просмотрели атрибуты у только что созданного файла и разрешили чтение и запись для категории пользователей «все остальные». Первоначально все группы имели право на чтение, а запись могли осуществлять все, кроме «остальных пользователей».

```
[guest@eliza ~]$ ls -l /tmp/file01.txt
ls: cannot access /tmp/file01.txt: No such file or directory
[guest@eliza ~]$ echo "test" > /tmp/file01.txt
[guest@eliza ~]$ ls -l /tmp/file01.txt
-rw-rw-r--. 1 guest guest 5 Oct  7 22:48 /tmp/file01.txt
[guest@eliza ~]$ chmod o+rw /tmp/file01.txt
[guest@eliza ~]$ ls -l /tmp/file01.txt
-rw-rw-rw-. 1 guest guest 5 Oct  7 22:48 /tmp/file01.txt
[guest@eliza ~]$
```

**Figure 15:** проверка прав на файле `file01.txt`

## От пользователя (не являющегося владельцем) попробовали прочитать файл

/file01.txt. Мы видим, что в файле записано одно слово. далее к нему мы должны были поочередно дозаписать слова test2 и test3, но я забыла добавить по одной галочке и вместо дозаписи перезаписала слова. От пользователя попробовали удалить файл /tmp/file01.txt командой rm /tmp/file01.txt, однако получила отказ.

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

## От суперпользователя командой выполнили команду, снимающую атрибут t

(Sticky-бит) с директории /tmp. Покинули режим суперпользователя командой exit. От пользователя проверили, что атрибута t у директории /tmp нет. Повторили предыдущие шаги. Получилось удалить файл.

```
[quest2@eliza tmp]$ su
Password:
[root@eliza tmp]# chmod -t /tmp
[root@eliza tmp]# exit
exit
[quest2@eliza tmp]$ ls -l / | grep tmp
drwxrwxrwx. 39 root root 4096 Oct  7 22:53 tmp
[root@eliza tmp]# echo "test2" >> /tmp/file01.txt
[root@eliza tmp]# cat file01.txt
test3
test2
[root@eliza tmp]# rm file01.txt
rm: remove regular file 'file01.txt'?
[root@eliza tmp]#
```

## **Выводы**

---

Изучили механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получили практические навыки работы в консоли с дополнительными атрибутами. Также мы рассмотрели работу механизма смены идентификатора процессов пользователей и влияние бита Sticky на запись и удаление файлов.