

## Лабораторная работа №5: Презентация.

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов.

---

Евдокимов Максим Михайлович. Группа - НФИбд-01-20.<sup>1</sup>

14 сентября, 2023, Москва, Россия

<sup>1</sup>Российский Университет Дружбы Народов

## Цели и задачи 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.

1. Изучить основы и особенности компилирование программ на Linux, работа с gcc, понятие объектный файл и другие.
2. Изучение механизмов изменения идентификаторов SetUID, SetGID и Sticky-битов.
3. Исследование Sticky-бита и рассмотрение его принципов работы в случае двух пользователей.

## Теория

---

Командой “gcc -v” проверяем наличие gcc и если нет то используем команду “yum install gcc”. Также чтобы менять атрибуты в системе уберём встроенную защиту от их изменений - SELinux, при помощи команды “setenforce 0”. И проверив что она работает командой “getenforce”, которая должна вывести: Permissive.

Компиляторы, доступные в Linux-системах, являются частью коллекции GNU-компиляторов, известной как GCC (GNU Compiler Collection, подробнее см. <http://gcc.gnu.org>). В неё входят компиляторы языков C, C++, Java, Objective-C, Fortran и Chill. Будем использовать лишь первые два.

Так из наличие можно проверить двумя командами: “whereis gcc” и “whereis g++”. Также в ходе работы будет создаваться - объектные файлы, которые невозможно запускать и использовать, поэтому после компиляции для получения готовой программы объектные файлы необходимо скомпоновать (автоудалить). Поэтому в ходе работы мы будем использовать команду “gcc <sub>файл</sub>.c -o <sub>файл</sub>” (где <sub>файл</sub> это имя файла кода с расширением .c), она не оставляет объектных файлов и даёт готовый к запуску файл.

## Выполнение лабораторной работы

---



Выполняем подготовку перед тем как начать работать, проверив наличие необходимых программ и их местоположение.

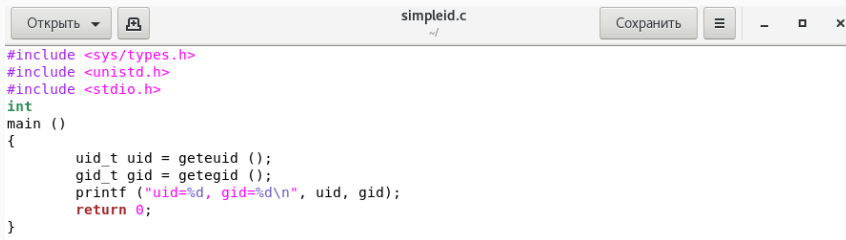
```
[max@Max ~]$ gcc -v
Используются внутренние спецификации.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/4.8.5/lto-wrapper
Целевая архитектура: x86_64-redhat-linux
Параметры конфигурации: ../configure --prefix=/usr --mandir=/usr/share/man --inf
odir=/usr/share/info --with-bugurl=http://bugzilla.redhat.com/bugzilla --enable-
bootstrap --enable-shared --enable-threads=posix --enable-checking=release --wit
h-system-zlib --enable-__cxa_atexit --disable-libunwind-exceptions --enable-gnu-
unique-object --enable-linker-build-id --with-linker-hash-style=gnu --enable-lan
guages=c,c++,objc,obj-c++,java,fortran,ada,go,lto --enable-plugin --enable-initf
ini-array --disable-libgcj --with-isl=/builddir/build/BUILD/gcc-4.8.5-20150702/o
bj-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
Модель многопоточности: posix
gcc версия 4.8.5 20150623 (Red Hat 4.8.5-44) (GCC)
[max@Max ~]$ sudo setenforce 0
[sudo] пароль для max:
[max@Max ~]$ getenforce
Permissive
[max@Max ~]$ whereis gcc
gcc: /usr/bin/gcc /usr/lib/gcc /usr/libexec/gcc /usr/share/man/man1/gcc.1.gz
[max@Max ~]$ whereis g++
g++: /usr/bin/g++ /usr/share/man/man1/g++.1.gz
```

Войдем в систему от имени пользователя guest.

```
[max@Max ~]$ su guest
Пароль:
[guest@Max max]$ cd
[guest@Max ~]$ touch simpleid.c
```

Рис. 2: Вход в систему

Создаём программу simpleid.c и вводим указанный код.



```
#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. 3: Код simpleid

Скомпилируем программу командой “gcc simpleid.c -o simpleid” и убедимся, что файл программы создан просмотрев папку.

```
пароль.  
[guest@Max max]$ cd  
[guest@Max ~]$ touch simpleid.c  
[guest@Max ~]$ gcc simpleid.c -o simpleid
```

Рис. 4: Компиляция и проверка

Выполните программу simpleid командой “./simpleid”.

```
[guest@Max ~]$ gcc simpleid.c -o simpleid  
[guest@Max ~]$ ./simpleid  
uid=1001, gid=1001
```

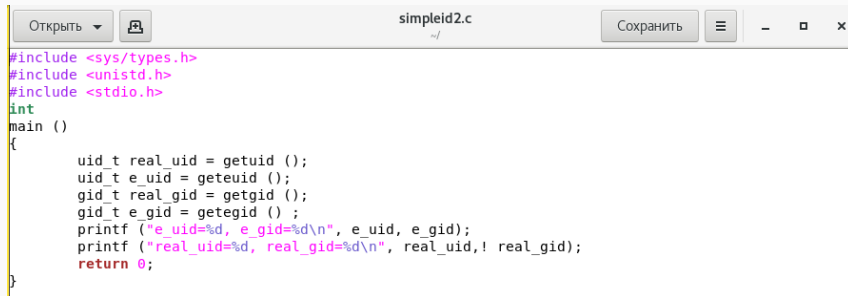
Рис. 5: Запуск программы simpleid

Выполним системную программу “id” и сравним полученный вами результат с данными предыдущего пункта задания.

```
[guest@Max ~]$ ./simpleid
uid=1001, gid=1001
[guest@Max ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined
_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 6: Проверка через id

Усложняем программу, добавив вывод действительных идентификаторов. Для этого создадим новый файл командой “touch 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;
}
```

Рис. 7: Код simpleid2

Скомпилируйте и запустите simpleid2.c командами “gcc simpleid2.c -o simpleid2” и “./simpleid2”.

```
[guest@Max ~]$ gcc simpleid2.c -o simpleid2  
[guest@Max ~]$ ./simpleid2  
e_uid=1001, e_gid=1001  
real_uid=1001, real_gid=0
```

Рис. 8: Компиляция и запуск



От имени суперпользователя выполним команды “sudo chown root:guest /home/guest/simpleid2” и “sudo chmod u+s /home/guest/simpleid2”.

```
[max@Max ~]$ sudo chown root:guest /home/guest/simpleid2  
[sudo] пароль для max:  
[max@Max ~]$ sudo chmod u+s /home/guest/simpleid2
```

Рис. 9: Изменяем UID файла

## Пункт 1.9

Используя `sudo` или повысив временно свои права с помощью `su`. Поясним, что делают эти команды.

```
[guest@Max ~]$ ls -la
итого 56
drwxrwx---. 7 guest guest 264 сен 18 13:05 .
drwxr-xr-x. 5 root  root  44 сен 11 20:56 ..
-rw-----. 1 guest guest 1790 сен 13 13:22 .bash_history
-rw-r--r--. 1 guest guest  18 апр  1 2020 .bash_logout
-rw-r--r--. 1 guest guest 193 апр  1 2020 .bash_profile
-rw-r--r--. 1 guest guest 231 апр  1 2020 .bashrc
drwxrwxr-x. 3 guest guest  18 сен  9 17:22 .cache
drwxrwxr-x. 6 guest guest  62 сен 17 22:14 .config
drwx-----. 2 guest guest  19 сен 13 11:25 dir1
drwxrwxr-x. 3 guest guest  19 сен 17 22:13 .local
drwxr-xr-x. 4 guest guest  39 сен  6 20:25 .mozilla
-rwxrwxr-x. 1 guest guest 8472 сен 18 13:02 simpleid
-rwxrwxr-x. 1 root  guest 8576 сен 18 13:05 simpleid2
-rw-rw-r--. 1 guest guest  311 сен 18 13:05 simpleid2.c
-rw-rw-r--. 1 guest guest  179 сен 18 13:01 simpleid.c
-rw-----. 1 guest guest  120 сен 17 22:14 .xauthbPs0tj
-rw-----. 1 guest guest  120 сен 17 20:39 .xauthTCzRqx
```

Выполним проверку правильности установки новых атрибутов и смены владельца файла simpleid2 командой “ls -l simpleid2”.

```
[guest@Max ~]$ ls -la simpleid2
-rwsrwxr-x. 1 root guest 8576 сен 18 13:05 simpleid2
[guest@Max ~]$ lsattr simpleid2
----- simpleid2
[guest@Max ~]$ ls -a
.                .bash_profile  dir1             simpleid2        .xauthTCzRqx
..               .bashrc        .local           simpleid2.c
.bash_history    .cache         .mozilla        simpleid.c
.bash_logout    .config        simpleid         .xauthbPs0tj
[guest@Max ~]$ ls -l
итого 32
drwx-----. 2 guest guest  19 сен 13 11:25 dir1
-rwxrwxr-x. 1 guest guest 8472 сен 18 13:02 simpleid
-rwsrwxr-x. 1 root  guest 8576 сен 18 13:05 simpleid2
-rw-rw-r--. 1 guest guest  311 сен 18 13:05 simpleid2.c
-rw-rw-r--. 1 guest guest  179 сен 18 13:01 simpleid.c
[guest@Max ~]$ ls -l simpleid2
-rwsrwxr-x. 1 root guest 8576 сен 18 13:05 simpleid2
```

Запустим simpleid2 и id командами “./simpleid2” и “id”. Сравним результаты.

```
[guest@Max ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=0
[guest@Max ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfine
_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

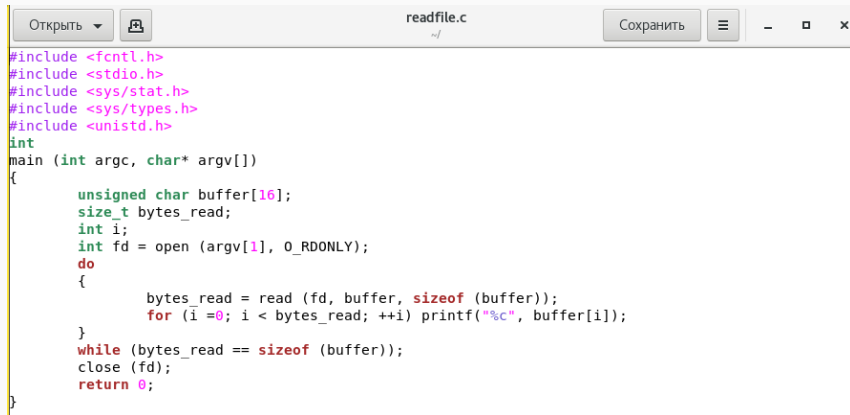
Рис. 12: Запуск программы simpleid2

Прделаем тоже самое относительно SetGID-бита командой “sudo chmod g+s /home/guest/simpleid2”.

```
[max@Max ~]$ sudo chmod u+s /home/guest/simpleid2  
[max@Max ~]$ sudo chmod g+s /home/guest/simpleid2
```

Рис. 13: Меняем GID файла

Создаём 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;
}
```

Рис. 14: Код readfile.c

Откомпилируем код выше ранее используемыми командами.

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

Рис. 15: Компиляция readfile.c

Сменим владельца у файла readfile.c (или любого другого текстового файла в системе) и изменим права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог. (выполнено не в полной мере)

```
[max@Max ~]$ sudo chown root:guest /home/guest/readfile  
[max@Max ~]$ sudo chmod 770 /home/guest/readfile  
[max@Max ~]$ sudo chmod 000 /home/guest/readfile
```

Рис. 16: Изменение владельца readfile



Проверим, что пользователь guest не может прочитать файл readfile.c.

```
[guest@Max ~]$ cat /home/guest/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;
}
```

Сменим у программы readfile владельца и установите SetU'D-бит.

```
[max@Max ~]$ sudo chmod u+s /home/guest/readfile
```

Рис. 18: Смена пользователя readfile

Проверим, может ли программа readfile прочитать файл readfile.c?

```
[guest@Max ~]$ ls -l readfile
-----. 1 root guest 8512 сен 18 13:22 readfile
[guest@Max ~]$ ls -l readfile
---S-----. 1 root guest 8512 сен 18 13:22 readfile
[guest@Max ~]$ ./readfile readfile.c
bash: ./readfile: Отказано в доступе
```

Рис. 19: Проверка 1 работы программы readfile

Проверим, может ли программа readfile прочитать файл /etc/shadow? Отразим полученный результат и ваши объяснения в отчёте

```
[guest@Max ~]$ ls -l readfile
-----. 1 root guest 8512 сен 18 13:22 readfile
[guest@Max ~]$ ls -l readfile
---S-----. 1 root guest 8512 сен 18 13:22 readfile
[guest@Max ~]$ ./readfile readfile.c
bash: ./readfile: Отказано в доступе
[guest@Max ~]$ ./readfile /etc/shadow
bash: ./readfile: Отказано в доступе
```

Рис. 20: Проверка 2 работы программы readfile

## Исследование Sticky-бита

---

Выясняем, установлен ли атрибут Sticky на директории /tmp, для чего выполните команду “ls -l / | grep tmp”.

```
[max@Max ~]$ ls -l / | grep tmp  
drwxrwxrwt. 22 root root 4096 сен 18 13:22 tmp  
[max@Max ~]$
```

Рис. 21: Смотрим атрибуты директории /tmp

От имени пользователя guest создадим файл file01.txt в директории /tmp со словом test командой “echo”test” > /tmp/file01.txt”.

```
[guest@Max ~]$ echo "test" > /tmp/file01.txt  
[guest@Max ~]$ ls -l /tmp/file01.txt  
-rw-rw-r--. 1 guest guest 5 сен 18 13:29 /tmp/file01.txt
```

Рис. 22: Создание файла file01.txt

Просмотрите атрибуты у только что созданного файла и разрешите чтение и запись для категории пользователей «все остальные» команды “ls -l /tmp/file01.txt”, “chmod o+rw /tmp/file01.txt” и “ls -l /tmp/file01.txt”.

```
[guest@Max ~]$ ls -l /tmp/file01.txt
-rw-rw-r--. 1 guest guest 5 сен 18 13:29 /tmp/file01.txt
[guest@Max ~]$ chmod o+rw /tmp/file01.txt
[guest@Max ~]$ ls -l /tmp/file01.txt
-rw-rw-rw-. 1 guest guest 5 сен 18 13:29 /tmp/file01.txt
```

Рис. 23: Изменение прав доступа



От пользователя guest2 (не являющегося владельцем) попробуем прочитать файл командой “cat /tmp/file01.txt”.

```
[guest2@Max ~]$ cat /tmp/file01.txt  
test
```

Рис. 24: Проверяем содержимое файла

От пользователя guest2 дозаписываем в файл /tmp/file01.txt слово test2 командой “echo”test2” > /tmp/file01.txt”. Что проходит успешно.

```
[guest2@Max ~]$ cat /tmp/file01.txt  
test  
[guest2@Max ~]$ echo "test2" > /tmp/file01.txt
```

Рис. 25: Изменяем содержание файла

Проверим содержимое файла командой “cat /tmp/file01.txt”.

```
[guest2@Max ~]$ echo "test2" > /tmp/file01.txt  
[guest2@Max ~]$ cat /tmp/file01.txt  
test2
```

Рис. 26: Проверяем содержимое файла

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"`.

```
[guest2@Max ~]$ echo "test2" > /tmp/file01.txt
[guest2@Max ~]$ cat /tmp/file01.txt
test2
[guest2@Max ~]$ echo "test3" > /tmp/file01.txt
```

Рис. 27: Изменяем содержание файла

Проверим содержимое файла командой “cat /tmp/file01.txt”.

```
[guest2@Max ~]$ echo "test3" > /tmp/file01.txt  
[guest2@Max ~]$ cat /tmp/file01.txt  
test3
```

Рис. 28: Проверяем содержимое файла

От пользователя guest2 попробуем удалить файл /tmp/file01.txt командой “rm /tmp/file01.txt”.

```
[guest2@Max ~]$ rm /tmp/file01.txt  
rm: невозможно удалить «/tmp/file01.txt»: Операция не позволена
```

Рис. 29: Пробуем удалить файл

Повысим свои права до суперпользователя командой “su -” и выполните после этого команду, снимающую атрибут t (Sticky-бит) “chmod -t /tmp”.

```
su: Сбой при проверке подлинности
[guest2@Max ~]$ su -
Пароль:
Последняя неудачная попытка входа в систему: Пн сен 18 13:34:01 MSK 202
Зна pts/2
Число неудачных попыток со времени последнего входа: 3.
[root@Max ~]# chmod -t /tmp
```

Рис. 30: меняем атрибут Sticky-бита

Покиньте режим суперпользователя командой “exit”.

```
[root@Max ~]# chmod -t /tmp
[root@Max ~]# ls -l /tmp/file01.txt
-rw-rw-rw-. 1 guest guest 6 сен 18 13:32 /tmp/file01.txt
[root@Max ~]# exit
logout
```

Рис. 31: Выход из суперпользователя



От пользователя guest2 проверим, что атрибута t у директории /tmp нет: “ls -l / | grep tmp”.

```
logout  
[guest2@Max ~]$ ls -l / | grep tmp  
drwxrwxrwx.  22 root root 4096 сен 18 13:34 tmp
```

Рис. 32: Проверка атрибутов

Повторив предыдущие шаги, можем заметить что мы теперь можем взаимодействовать с файлом как раньше и способны его удалить.

```
[guest2@Max ~]$ cat /tmp/file01.txt
test3
[guest2@Max ~]$ echo "test1" > /tmp/file01.txt
[guest2@Max ~]$ cat /tmp/file01.txt
test1
[guest2@Max ~]$ rm /tmp/file01.txt
rm: невозможно удалить «/tmp/file01.txt»: Нет такого файла или каталог
а
[guest2@Max ~]$ rm /tmp/file01.txt
```

Рис. 33: Повтор шагов

Удалось удалить файл от имени пользователя, не являющегося его владельцем.

```
[guest2@Max ~]$ ls -l /tmp/
итого 228
drwxr-xr-x. 2 root root    18 сен  9 22:20 hsuperfdata_root
-rw-r--r--. 1 root root    71 сен  9 22:18 lua_bSaede
drwx-----. 2 max max     24 сен 17 20:35 ssh-RLjzGYf0pkBL
drwx-----. 2 max max     24 сен 14 15:02 ssh-sCsZfamsvb9d
drwx-----. 3 root root    17 сен 17 20:35 systemd-private-b84ab87994
df44598649f93d7dde2401-bolt.service-07akIv
drwx-----. 3 root root    17 сен 17 20:35 systemd-private-b84ab87994
df44598649f93d7dde2401-colord.service-zZaL0s
drwx-----. 3 root root    17 сен 17 20:35 systemd-private-b84ab87994
df44598649f93d7dde2401-cups.service-v0qv3J
drwx-----. 3 root root    17 сен 17 20:36 systemd-private-b84ab87994
df44598649f93d7dde2401-fwupd.service-KHmUoS
drwx-----. 3 root root    17 сен 17 20:35 systemd-private-b84ab87994
df44598649f93d7dde2401-rtkit-daemon.service-pkmlEd
drwx-----. 3 root root    17 сен 14 13:41 systemd-private-f13578ce95
204621b151e2e29be0a591-bolt.service-c58yYz
drwx-----. 3 root root    17 сен 14 13:41 systemd-private-f13578ce95
204621b151e2e29be0a591-colord.service-CI73XR
drwx-----. 3 root root    17 сен 14 13:41 systemd-private-f13578ce95
204621b151e2e29be0a591-cups.service-w2mqld
drwx-----. 3 root root    17 сен 14 15:02 systemd-private-f13578ce95
204621b151e2e29be0a591-fwupd.service-Hu4gb0
drwx-----. 3 root root    17 сен 14 13:41 systemd-private-f13578ce95
204621b151e2e29be0a591-rtkit-daemon.service-2bJ2I0
drwx-----. 2 max max      6 сен 18 13:53 tracker-extract-files.1000
-rw-r--r--. 1 root root 27718 сен  9 22:24 vboxguest-Module.symvers
-rw-----. 1 root root 199861 сен  9 10:58 yum_save_tx.2023-09-09.10-
58.cIvB8i.yumtx
```

Повысим свои права до суперпользователя и вернём атрибут `t` на директорию выполнив цепочку команд “`su -`”, “`chmod +t /tmp`” и “`exit`”.

```
[guest2@Max ~]$ su -  
Пароль:  
Последний вход в систему: Пн сен 18 13:34:11 MSK 2023 на pts/2  
[root@Max ~]# chmod +t /tmp  
[root@Max ~]# exit  
_
```

Рис. 35: Возвращаем все изменения

## Выводы по проделанной работе

---

Изучены механизмы изменения идентификаторов UID, GID и Sticky-битов. Получены практические навыки работы в консоли с дополнительными атрибутами. Рассмотрены работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.