

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Лабораторная работа № 5

Абу Сувейлим М. М.

10 января 2003

Российский университет дружбы народов, Москва, Россия

Информация

- Абу Сувейлим Мухаммед Мунифович
- Студент
- Российский университет дружбы народов
- 1032215135@pfur.ru
- <https://mukhammed-abu-suveilim.github.io/>

Вводная часть

- 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.
- Создание программы и исследование Sticky-бита.

1. // skillbox.ru.
2. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.

Выполнение лабораторной работы

Проверяем, что у нас установлен компилятор gcc командой (рис. 1):

```
[root@node snabu]# gcc -v
Используется внутреннее специфическое.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/11/lto-wrapper
OFFLOAD_TARGET_NAMES=none
OFFLOAD_TARGET_DEFAULT=1
Целевая архитектура: x86_64-redhat-linux
Параметры конфигурации: ../configure --enable-bootstrap --enable-host-pie --enable-host-shared --enable-languages=c,c++,fortran,lto --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --with-bugurl=http://bugzilla.redhat.com/bugzilla --enable-shared --enable-threads=posix --enable-checking=release --enable-system-zlib --enable-_cxa_atexit --disable-libunwind-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-gcc-major-version-only --enable-plugin --enable-lto --enable-lto-flags --without-lto --enable-multilib --with-linker-hash-style=gnu --enable-offload-targets=none --without-cuda-driver --enable-gnu-indirect-function --enable-cet --with-tune=generic --with-arch_64=x86-64-v2 --with-arch_32=x86-64 --build=x86_64-redhat-linux --with-build-config=bootstrap-lto --enable-link-serialization=1
Модель адресности: posix
Supported LTO compression algorithms: zlib zstd
gcc version 11.5.0 20240719 (Red Hat 11.5.0-2) (GCC)
[root@node snabu]#
```

Figure 1: Компилятор gcc

Далее, отключим систему запретов до очередной перезагрузки системы командой (рис. 2):

```
[root@smabu smabu]# setenforce 0  
[root@smabu smabu]# getenforce  
Permissive  
[root@smabu smabu]#
```

Figure 2: Команда setenforce

Войдем в систему от имени пользователя guest и создадим программу simpleid.c (рис. 3):



```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 int
5 main ()
6
7     uid_t uid = geteuid ();
8     gid_t gid = getegid ();
9     printf ("uid=%d, gid=%d\n", uid, gid);
10     return 0;
```

Figure 3: Программа simpleid.c

Скомпилируем программу и убедимся, что файл программы создан (рис. 4):

```
[guest@smabu lab05]$ touch simpleid.c
[guest@smabu lab05]$ ls
simpleid.c
[guest@smabu lab05]$ gcc simpleid.c -o simpleid
[guest@smabu lab05]$ ./simpleid
uid=1001, gid=1001
[guest@smabu lab05]$ id
uid=1001(guest) gid=1001(guest) rгруппы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@smabu lab05]$
```

Figure 4: Запуск программы simpleid.c

Выполним программу simpleid и системную программу id. Видем, что полученный нами результат с данными предыдущего пункта задания и этого пункта задания совпадают.

Усложним программу, добавив вывод действительных идентификаторов (рис. 5):

```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 int
5 main ()
6 {
7     uid_t real_uid = getuid ();
8     uid_t e_uid = geteuid ();
9     gid_t real_gid = getgid ();
10    gid_t e_gid = getegid ();
11    printf ("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
12    printf ("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
13    return 0;
14 }
```

Figure 5: Программа simpleid2.c

Скомпилируем и запустим simpleid2.c (рис. 6):

```
guest@smabu lab05]$ ls -l simpleid2
-rwsr-xr-x. 1 root guest 17656 окт  5 13:08 simpleid2
guest@smabu lab05]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
guest@smabu lab05]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
guest@smabu lab05]$
```

Figure 6: Запуск программы simpleid2.c

От имени суперпользователя выполним команды:

```
chown root:guest /home/guest/simpleid2
```

```
chmod u+s /home/guest/simpleid2
```

Проверка правильности установки новых атрибутов

Выполним проверку правильности установки новых атрибутов и смены владельца файла `simpleid2` и запустим `simpleid2` и `id` (рис. 7):

```
guest@smabu lab05]$ ls -l simpleid2
-rwsr-xr-x. 1 root guest 17656 окт  5 13:08 simpleid2
guest@smabu lab05]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
guest@smabu lab05]$ id
uid=1001(guest) gid=1001(guest) rpyнны=1001(guest) контекст=unconfined_u:uncon
fined_r:unconfined_t:s0-s0:c0.c1023
guest@smabu lab05]$
```

Figure 7: Проверка правильности установки новых атрибутов

Создаим программу readfile.c (рис. 8):

```
GNU nano 5.6.1
#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;
}
```

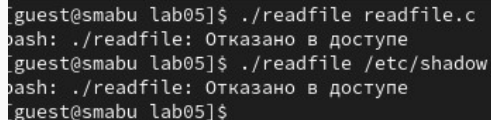
Figure 8: Программа readfile.c

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. (рис. 9):

```
[guest@smabu lab05]$ su
Пароль:
[root@smabu lab05]# chmod u+s /home/guest/lab05/readfile.c
[root@smabu lab05]# chmod u+s /home/guest/lab05/readfile.c
[root@smabu lab05]#
```

Figure 9: Смена владельца у файла readfile.c

Программа readfile не может прочитать файл readfile.c и не может и прочесть файл /etc/shadow (рис. 10)



```
[guest@smabu lab05]$ ./readfile readfile.c
bash: ./readfile: Отказано в доступе
[guest@smabu lab05]$ ./readfile /etc/shadow
bash: ./readfile: Отказано в доступе
[guest@smabu lab05]$
```

Figure 10: Чтения файла readfile.c

Выясним, установлен ли атрибут Sticky на директории /tmp, для чего выполним команду (рис. 11):

```
[guest@smabu lab05]$ ls -l / | grep tmp  
drwxrwxrwt. 19 root root 4096 окт  5 13:30 tmp  
[guest@smabu lab05]$
```

Figure 11: Команда grep

От имени пользователя guest создадим файл file01.txt в директории /tmp со словом test (рис. 12):

```
[guest@smabu lab05]$ ls -l / | grep tmp
drwxrwxrwt. 19 root root 4096 окт  5 13:30 tmp
[guest@smabu lab05]$ echo "test" > /tmp/file01.txt
[guest@smabu lab05]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 окт  5 13:32 /tmp/file01.txt
[guest@smabu lab05]$ chmod o_rw /tmp/file01.txt
chmod: неверный режим: «o_rw»
По команде «chmod --help» можно получить дополнительную информацию.
[guest@smabu lab05]$ chmod o+rw /tmp/file01.txt
[guest@smabu lab05]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 окт  5 13:32 /tmp/file01.txt
[guest@smabu lab05]$
```

Figure 12: Файл file01.txt

Просмотрим атрибуты у только что созданного файла и разрешите чтение и запись для категории пользователей «все остальные» (рис. 12).

Чтения файла file01.txt от пользователя guest2

От пользователя guest2 (не являющегося владельцем) попробуем прочитать файл /tmp/file01.txt (рис. 13):

```
[guest@smabu lab05]$ su - guest2
Пароль:
[guest2@smabu ~]$ whoami
guest2
[guest2@smabu ~]$ echo "test2" > /tmp/file01.txt
-bash: /tmp/file01.txt: Отказано в доступе
[guest2@smabu ~]$ cat /tmp/file01.txt
test
[guest2@smabu ~]$ echo "test3" > /tmp/file01.txt
-bash: /tmp/file01.txt: Отказано в доступе
[guest2@smabu ~]$ cat /tmp/file01.txt
test
[guest2@smabu ~]$ rm /tmp/file01.txt
rm: удалить защищённый от записи обычный файл '/tmp/file01.txt'? y
rm: невозможно удалить '/tmp/file01.txt': Операция не позволена
[guest2@smabu ~]$
```

Figure 13: Чтения файла file01.txt от пользователя guest2

Попробуем дозаписать в файл `/tmp/file01.txt` слово `test2` командой (рис. 13). Проверим содержимое файла командой (рис. 13). От пользователя `guest2` попробуем записать в файл `/tmp/file01.txt` слово `test3`, стерев при этом всю имеющуюся в файле информацию командой (рис. 13). Проверим содержимое файла и попробуем удалить файл `/tmp/file01.txt` командой (рис. 13).

Выполним после этого команду, снимающую атрибут `t` (Sticky-бит) с директории `/tmp` (рис. 14):

```
[guest2@smabu ~]$ su
Пароль:
[root@smabu guest2]# chmod -t /tmp
[root@smabu guest2]# exit
exit
[guest2@smabu ~]$
```

Figure 14: Команда `chmod -t /tmp`

Команда `chmod -t /tmp`

От пользователя `guest2` проверим, что атрибута `t` у директории `/tmp` нет (рис. 15):

```
Пароль:
[root@smabu guest2]# chmod -t /tmp
[root@smabu guest2]# exit
exit
[guest2@smabu ~]$ ls -l | grep tmp
[guest2@smabu ~]$ su
Пароль:
su: Сбой при проверке подлинности
[guest2@smabu ~]$ su
Пароль:
[root@smabu guest2]# chmod +t /tmp
[root@smabu guest2]# exit
exit
[guest2@smabu ~]$ ls -l | grep tmp
[guest2@smabu ~]$
```

Figure 15: Команда `chmod -t /tmp`

Выводы

Изучали механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получили практические навыки работы в консоли с дополнительными атрибутами. Рассмотрели работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов кратко описываются итоги проделанной работы.