

Лабораторная работа №5

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

Дарья Эдуардовна Ибатулина

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Подготовка лабораторного стенда	9
4.2	Создание программы	10
4.3	Исследование Sticky-бита	16
5	Выводы	18
	Список литературы	19

Список иллюстраций

4.1	Проверка установки gcc	9
4.2	Установка setenforce, проверка getenforce	9
4.3	Создание файла simpleid.c	10
4.4	Файл simpleid.c	10
4.5	Компиляция, исполнение файла simpleid.c, сравнение результата с выводом команды id	10
4.6	Файл simpleid2.c	11
4.7	Запуск программы simpleid2.c	11
4.8	Смена владельца файла и передача некоторых прав пользователю guest	12
4.9	Проверка правильности установки атрибутов	13
4.10	Файл readfile.c	13
4.11	Установка прав на файл readfile.c	13
4.12	Проверка правильности выполненных действий	15
4.13	Выполнение команд от имени суперпользователя	15
4.14	Выполнение команд от имени суперпользователя	16
4.15	Выполнение заданий	16
4.16	Выполнение заданий	17
4.17	Выполнение заданий	17

Список таблиц

1 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.

2 Задание

1. Подготовить лабораторный стенд;
2. Поработать с компилятором gcc;
3. Попрактиковаться в установке и снятии атрибутов с файлов.

3 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 относиться 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 `rw-` или `rwX`? 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]

4 Выполнение лабораторной работы

4.1 Подготовка лабораторного стенда

Проверяем, установлен ли gcc (рис. [4.1]).

```
[guest@deibatulina ~]$ gcc -v
Используются внутренние спецификации.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/11/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Целевая архитектура: x86_64-redhat-linux
Параметры конфигурации: ./configure --enable-bootstrap --enable-host-pie --enable-host-bind-now --enable-languages=c,c++,fortran,lto --prefix=/usr --mandir=/usr/share/man --infodir=/usr/share/info --with-bugurl=https://bugs.rockylinux.org/ --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-gcc-major-version-only --enable-plugin --enable-initfini-array --without-isl --enable-multilib --with-linker-hash-style=gnu --enable-offload-targets=nvptx-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 многопоточности: posix
Supported LTO compression algorithms: zlib zstd
gcc версия 11.4.1 20230605 (Red Hat 11.4.1-2) (GCC)
[guest@deibatulina ~]$
```

Рис. 4.1: Проверка установки gcc

Устанавливаем setenforce 0 и проверяем что getenforce выдает: “Permissive” (рис. [4.2]).

```
[deibatulina@deibatulina ~]$ sudo -i
[sudo] пароль для deibatulina:
[root@deibatulina ~]# setenforce 0
[root@deibatulina ~]# getenforce
Permissive
[root@deibatulina ~]#
```

Рис. 4.2: Установка setenforce, проверка getenforce

4.2 Создание программы

Переключаемся на учетную запись администратора и создаем файл `simpleid.c`, заполняем его предложенной программой (рис. [4.3], [4.4]).

```
[root@deibatulina ~]# su - guest
[guest@deibatulina ~]$ touch simpleid.c
```

Рис. 4.3: Создание файла `simpleid.c`



```
GNU nano 5.6.1 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. 4.4: Файл `simpleid.c`

Скомпилируем программу, выполним её и сравним вывод команды `id` с результатом исполнения программы. Они совпадают (рис. [4.5]).

```
[guest@deibatulina ~]$ gcc simpleid.c -o simpleid
[guest@deibatulina ~]$ ls
dir1 simpleid simpleid.c
[guest@deibatulina ~]$ ./simpleid
uid=1001, gid=1001
[guest@deibatulina ~]$ id
uid=1001(guest) gid=1001(guest) rpyнны=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@deibatulina ~]$
```

Рис. 4.5: Компиляция, исполнение файла `simpleid.c`, сравнение результата с выводом команды `id`

Листинг программы файла `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;
}

```

Создадим другой файл simpleid2.c, введём в него код более сложной программы, выполним его (рис. [4.6], [4.7]).



```

GNU nano 5.6.1 simpleid.c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int
main ()
{
    uid_t real_uid = geteuid ();
    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. 4.6: Файл simpleid2.c



```

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

```

Рис. 4.7: Запуск программы simpleid2.c

Листинг программы файла simpleid2.c:

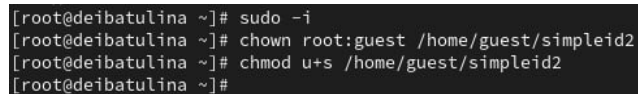
```

#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;
}

```

Переключаемся на администратора и выполняем следующие команды (рис. [4.8]).



```

[root@deibatulina ~]# sudo -i
[root@deibatulina ~]# chown root:guest /home/guest/simpleid2
[root@deibatulina ~]# chmod u+s /home/guest/simpleid2
[root@deibatulina ~]#

```

Рис. 4.8: Смена владельца файла и передача некоторых прав пользователю guest

Этими командами была произведена смена пользователя файла на root и установлен SetUID-бит.

Выполним проверку правильности установки атрибутов, а также скомпилируем и запустим файл simpleid2, сравним результат с выводом команды id (рис. [4.9]).

```
[root@deibatulina ~]# su - guest
[guest@deibatulina ~]$ ls -l simpleid2
-rwsr-xr-x. 1 root guest 26016 map 13 19:39 simpleid2
[guest@deibatulina ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=1001
[guest@deibatulina ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfi
ned_r:unconfined_t:s0-s0:c0.c1023
[guest@deibatulina ~]$
```

Рис. 4.9: Проверка правильности установки атрибутов

Создаём файл `readfile.c` (рис. [4.10]) и изменяем его владельца так, чтобы только суперпользователь мог прочитать его, а `guest` не мог (рис. [4.11]).



```
guest@deibatulina:~
GNU nano 5.6.1 readfile.c Изменён
#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));
```

Рис. 4.10: Файл `readfile.c`

```
[guest@deibatulina ~]$ su
Пароль:
[root@deibatulina guest]# chown root:guest readfile
[root@deibatulina guest]# chmod 700 readfile
[root@deibatulina guest]# chown root:guest readfile
[root@deibatulina guest]# chmod -r readfile.c
[root@deibatulina guest]# chmod u+c readfile
chmod: неверный режим: «u+c»
По команде «chmod --help» можно получить дополнительную информацию.
[root@deibatulina guest]# chmod u+s readfile
[root@deibatulina guest]#
```

Рис. 4.11: Установка прав на файл `readfile.c`

Листинг программы файла `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;
}

```

Проверим, что пользователь guest не может прочитать данный файл. Также программа readfile не сможет прочитать файл readfile.c и readfile не может прочитать файл /etc/shadow (рис. [4.12]).

```
[root@deibatulina guest]# exit
exit
[guest@deibatulina ~]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@deibatulina ~]$ ./readfile readfile.c
-bash: ./readfile: Отказано в доступе
[guest@deibatulina ~]$ ./readfile /etc/shadow
-bash: ./readfile: Отказано в доступе
[guest@deibatulina ~]$
```

Рис. 4.12: Проверка правильности выполненных действий

Но от имени суперпользователя удаётся выполнить все вышеприведённые команды (рис. [4.13], [4.14]).

```
[guest@deibatulina ~]$ su
Пароль:
[root@deibatulina 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@deibatulina guest]# ./readfile readfile.c
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
```

Рис. 4.13: Выполнение команд от имени суперпользователя

```
[root@deibatulina guest]# ./readfile /etc/shadow
root:$6$P/RvTs4KAaNPmH8$6UkaWihIs09VqnZEXEx4TgSPw9vktZ73bxSUGwpHRSeuUYLF5wAaWqq
hoisdwF4.4PaACux7leqyaALeEYtS.:0:99999:7:::
bin:!:19469:0:99999:7:::
daemon:!:19469:0:99999:7:::
adm:!:19469:0:99999:7:::
lp:!:19469:0:99999:7:::
sync:!:19469:0:99999:7:::
shutdown:!:19469:0:99999:7:::
halt:!:19469:0:99999:7:::
mail:!:19469:0:99999:7:::
operator:!:19469:0:99999:7:::
games:!:19469:0:99999:7:::
ftp:!:19469:0:99999:7:::
nobody:!:19469:0:99999:7:::
systemd-coredump:!:19767:!:19767:7:::
dbus:!:19767:!:19767:7:::
polkitd:!:19767:!:19767:7:::
avahi:!:19767:!:19767:7:::
rtkit:!:19767:!:19767:7:::
pipewire:!:19767:!:19767:7:::
sssd:!:19767:!:19767:7:::
libstoragemgmt:!:19767:!:19767:7:::
systemd-oom:!:19767:!:19767:7:::
tss:!:19767:!:19767:7:::
geoclue:!:19767:!:19767:7:::
cockpit-ws:!:19767:!:19767:7:::
cockpit-wsinstance:!:19767:!:19767:7:::
flatpak:!:19767:!:19767:7:::
colord:!:19767:!:19767:7:::
clevi:!:19767:!:19767:7:::
setroubleshoot:!:19767:!:19767:7:::
gdm:!:19767:!:19767:7:::
pesign:!:19767:!:19767:7:::
```

Рис. 4.14: Выполнение команд от имени суперпользователя

4.3 Исследование Sticky-бита

Выясним, установлен ли атрибут Sticky на директории /tmp, от имени пользователя guest создадим файл file01.txt в директории /tmp со словом test, и посмотрим атрибуты у только что созданного файла и разрешим чтение и запись для категории пользователей «все остальные» (рис. [4.15]).

```
[deibatulina@deibatulina ~]$ ls -l / | grep tmp
drwxrwxrwt. 15 root root 4096 map 13 19:58 tmp
[deibatulina@deibatulina ~]$ echo "test" > /tmp/file01.txt
[deibatulina@deibatulina ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 deibatulina deibatulina 5 map 13 20:02 /tmp/file01.txt
[deibatulina@deibatulina ~]$ chmod o+rw /tmp/file01.txt
[deibatulina@deibatulina ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 deibatulina deibatulina 5 map 13 20:02 /tmp/file01.txt
[deibatulina@deibatulina ~]$
```

Рис. 4.15: Выполнение заданий

От пользователя guest2 (не являющегося владельцем) попробуем прочитать

файл /tmp/file01.txt:, от пользователя guest2 попробуем дозаписать в файл /tmp/file01.txt слово test2, проверим содержимое файла, от пользователя guest2 попробуем записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию и проверим содержимое файла (рис. [4.16]).

```
[deibatulina@deibatulina ~]$ su - guest2
Пароль:
[guest2@deibatulina ~]$ cat /tmp/file01.txt
test
[guest2@deibatulina ~]$ echo "test2" > /tmp/file01.txt
[guest2@deibatulina ~]$ cat /tmp/file01.txt
test2
[guest2@deibatulina ~]$ echo "test3" > /tmp/file01.txt
[guest2@deibatulina ~]$ cat /tmp/file01.txt
test3
[guest2@deibatulina ~]$
```

Рис. 4.16: Выполнение заданий

Удалить данный файл от имени guest2 не удаётся.

Повысим свои права до суперпользователя и выполним после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp, а затем покинем режим суперпользователя. От пользователя guest2 проверим, что атрибута t у директории /tmp нет. Повторив предыдущие шаги от имени других пользователей, я могу заметить, что всё получается. Отвечая на вопрос: Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем? я могу сказать, что удалось. Повысим свои права до суперпользователя и вернём атрибут t на директорию /tmp (рис. [4.17]).

```
[guest2@deibatulina ~]$ su -
Пароль:
[root@deibatulina ~]# chmod -t /tmp
[root@deibatulina ~]# exit
выход
[guest2@deibatulina ~]$ ls -l / | grep tmp
drwxrwxrwx. 14 root root 4096 мар 13 20:08 tmp
[guest2@deibatulina ~]$ su -
Пароль:
[root@deibatulina ~]# chmod +t /tmp
[root@deibatulina ~]# exit
выход
[guest2@deibatulina ~]$
```

Рис. 4.17: Выполнение заданий

5 Выводы

Мною 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

[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>