

# **Отчёт по лабораторной работе №5**

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

Кирилл Захаров

# Содержание

|          |                                       |           |
|----------|---------------------------------------|-----------|
| <b>1</b> | <b>Цель работы</b>                    | <b>4</b>  |
| <b>2</b> | <b>Выполнение лабораторной работы</b> | <b>5</b>  |
| 2.1      | Подготовка . . . . .                  | 5         |
| 2.2      | Изучение механики SetUID . . . . .    | 6         |
| 2.3      | Исследование Sticky-бита . . . . .    | 10        |
| <b>3</b> | <b>Выводы</b>                         | <b>13</b> |
|          | <b>Список литературы</b>              | <b>14</b> |

# List of Figures

|     |   |    |
|-----|---|----|
| 2.1 | подготовка к работе . . . . .           | 5  |
| 2.2 | программа simpleid . . . . .            | 6  |
| 2.3 | результат программы simpleid . . . . .  | 6  |
| 2.4 | программа simpleid2 . . . . .           | 7  |
| 2.5 | результат программы simpleid2 . . . . . | 8  |
| 2.6 | программа readfile . . . . .            | 9  |
| 2.7 | результат программы readfile . . . . .  | 10 |
| 2.8 | исследование Sticky-бита . . . . .      | 12 |

# 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 Выполнение лабораторной работы

### 2.1 Подготовка

1. Для выполнения части заданий требуются средства разработки приложений. Проверили наличие установленного компилятора gcc командой `gcc -v`: компилятор обнаружен.
2. Чтобы система защиты SELinux не мешала выполнению заданий работы, отключили систему запретов до очередной перезагрузки системы командой `setenforce 0`:
3. Команда `getenforce` вывела `Permissive`:

```
root@kzaharov:/home/guest# gcc -v
Используя внутренние спецификации.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/14/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none:amdgc-n-amdhsa
OFFLOAD_TARGET_DEFAULT=1
Целевая архитектура: x86_64-redhat-linux
Параметры конфигурации: ../configure --enable-bootstrap --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-multilib --with-system-zlib --enable-_cxa_atexit --disable-libunwind-exceptions --enable-gnu-unique-object --enable-linker-build-id --with-gcc-major-version-only --enable-libstdcxx-backtrace --with-libstdcxx-zoneinfo=/usr/share/zoneinfo --with-linker-hash-style=gnu --enable-plugin --enable-initfini-array --without-isl --enable-offload-targets=nvptx-none,amdgc-n-amdhsa --enable-offload-defaulted --without-cuda-driver --enable-gnu-indirect-function --enable-cet --with-tune=generic --with-arch_64=x86-64-v3 --with-arch_32=x86-64 --build=x86_64-redhat-linux --with-build-config=bootstrap-lto --enable-link-serialization=1 --enable-host-pie --enable-host-bind-now
Модель многопоточности: posix
Supported LTO compression algorithms: zlib zstd
gcc версия 14.2.1 20250110 (Red Hat 14.2.1-7) (GCC)
root@kzaharov:/home/guest#
root@kzaharov:/home/guest# setenforce 0
root@kzaharov:/home/guest# getenforce
Permissive
root@kzaharov:/home/guest#
```

Figure 2.1: подготовка к работе

## 2.2 Изучение механики SetUID

1. Вошли в систему от имени пользователя guest.
2. Написали программу simpleid.c.



```
Открыть ▾ + simpleid.c
~/lab5

#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 2.2: программа simpleid


3. Скомпилировали программу и убедились, что файл программы создан: gcc simpleid.c -o simpleid
4. Выполнили программу simpleid командой ./simpleid
5. Выполнили системную программу id с помощью команды id. uid и gid совпадает в обеих программах



```
guest@kzaharov:~$ cd
guest@kzaharov:~$ mkdir lab5
guest@kzaharov:~$ cd lab5/
guest@kzaharov:~/lab5$ touch simpleid.c
guest@kzaharov:~/lab5$ gcc simpleid.c
guest@kzaharov:~/lab5$ gcc simpleid.c -o simpleid
guest@kzaharov:~/lab5$ ./simpleid
uid=1001, gid=1001
guest@kzaharov:~/lab5$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
guest@kzaharov:~/lab5$
```

Figure 2.3: результат программы simpleid

6. Усложнили программу, добавив вывод действительных идентификаторов.



```
Открыть ▾ + simpleid2.c
~/lab5

#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int main()
{
    uid_t e_uid = geteuid();
    gid_t e_gid = getegid();
    uid_t real_uid = getuid();
    gid_t real_gid = getgid();
    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 2.4: программа simpleid2

7. Скомпилировали и запустили simpleid2.c:

```
gcc simpleid2.c -o simpleid2
./simpleid2
```

8. От имени суперпользователя выполнили команды:

```
chown root:guest /home/guest/simpleid2
chmod u+s /home/guest/simpleid2
```

9. Использовали su для повышения прав до суперпользователя

10. Выполнили проверку правильности установки новых атрибутов и смены владельца файла simpleid2:

```
ls -l simpleid2
```

11. Запустили simpleid2 и id:

```
./simpleid2
```

```
id
```

Результат выполнения программ теперь немного отличается

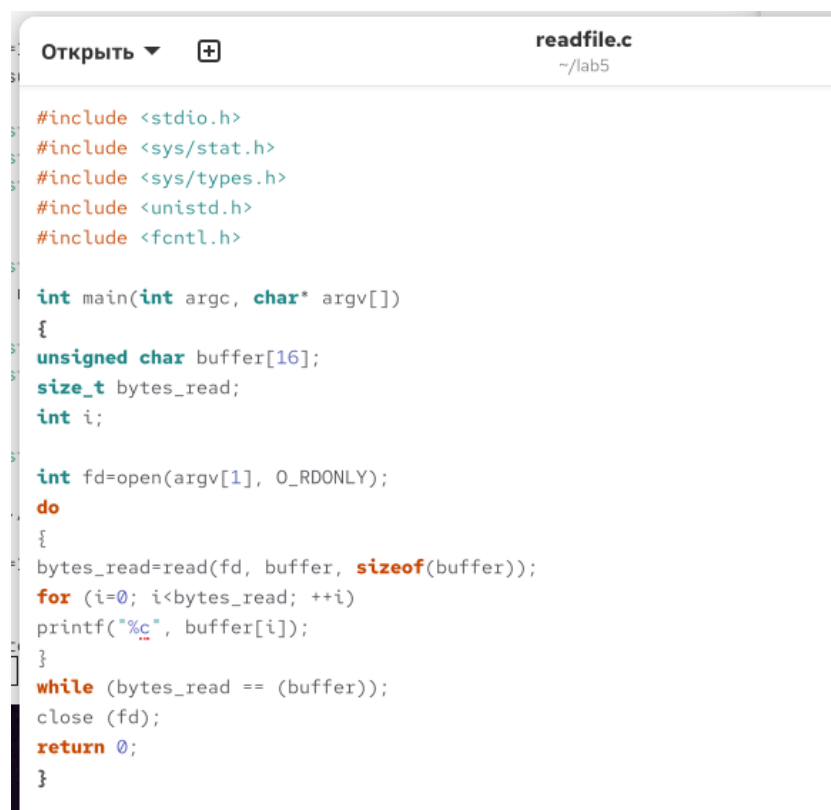
## 12. Проделали тоже самое относительно SetGID-бита.

```
guest@kzaharov:~/lab5$  
guest@kzaharov:~/lab5$ touch simpleid2.c  
guest@kzaharov:~/lab5$ gcc simpleid2.c  
guest@kzaharov:~/lab5$ gcc simpleid2.c -o simpleid2  
guest@kzaharov:~/lab5$ ./simpleid2  
e_uid=1001, e_gid=1001  
real_uid=1001, real_gid=1001  
guest@kzaharov:~/lab5$ su  
Пароль:  
root@kzaharov:/home/guest/lab5# chown root:guest simpleid2  
root@kzaharov:/home/guest/lab5# chmod u+s simpleid2  
root@kzaharov:/home/guest/lab5# ./simpleid2  
e_uid=0, e_gid=0  
real_uid=0, real_gid=0  
root@kzaharov:/home/guest/lab5# id  
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:s  
0-s0:c0.c1023  
root@kzaharov:/home/guest/lab5# chmod g+s simpleid2  
root@kzaharov:/home/guest/lab5# ./simpleid2  
e_uid=0, e_gid=1001  
real_uid=0, real_gid=0  
root@kzaharov:/home/guest/lab5#  
exit  
guest@kzaharov:~/lab5$ ./simpleid2  
e_uid=0, e_gid=1001  
real_uid=1001, real_gid=1001  
guest@kzaharov:~/lab5$ █
```

Figure 2.5: результат программы simpleid2

## 13. Написали программу readfile.c



The image shows a code editor window with the title 'readfile.c' and a file path '~./lab5'. The code is a C program that opens a file specified by the first command-line argument in read-only mode. It reads the file in 16-byte chunks into a buffer and prints each byte as a character. The code includes standard headers for file operations and string handling.

```
Открыть ▾ + readfile.c
~./lab5

#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.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 == (buffer));
    close (fd);
    return 0;
}
```

Figure 2.6: программа readfile

14. Откомпилировали её.

```
gcc readfile.c -o readfile
```

15. Сменили владельца у файла readfile.c и изменили права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог.

```
chown root:guest /home/guest/readfile.c
```

```
chmod 700 /home/guest/readfile.c
```

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

17. Сменили у программы readfile владельца и установили SetU'D-бит.

18. Проверили, может ли программа readfile прочитать файл readfile.c

19. Проверили, может ли программа readfile прочитать файл /etc/shadow

```

guest@kzaharov:~/lab5$ touch readfile.c
guest@kzaharov:~/lab5$ gcc readfile.c
readfile.c: В функции «main»:
readfile.c:20:19: предупреждение: сравнение указателя и целого
    20 | while (bytes_read == (buffer));
        |                      ^~
guest@kzaharov:~/lab5$ gcc readfile.c -o readfile
readfile.c: В функции «main»:
readfile.c:20:19: предупреждение: сравнение указателя и целого
    20 | while (bytes_read == (buffer));
        |                      ^~
guest@kzaharov:~/lab5$ su
Пароль:
root@kzaharov:/home/guest/lab5# chown root:root readfile
root@kzaharov:/home/guest/lab5# chmod -rwx readfile.c
root@kzaharov:/home/guest/lab5# chmod u+s readfile
root@kzaharov:/home/guest/lab5#
exit
guest@kzaharov:~/lab5$ cat readfile.c
cat: readfile.c: Отказано в доступе
guest@kzaharov:~/lab5$
guest@kzaharov:~/lab5$ ./readfile readfile.c
#include <stdio.h>
guest@kzaharov:~/lab5$
guest@kzaharov:~/lab5$ ./readfile /etc/shadow
root:$y$j9T$z1ZFguest@kzaharov:~/lab5$
guest@kzaharov:~/lab5$

```

Figure 2.7: результат программы readfile

## 2.3 Исследование Sticky-бита

1. Выяснили, установлен ли атрибут Sticky на директории /tmp:

```
ls -l / | grep tmp
```

2. От имени пользователя guest создали файл file01.txt в директории /tmp со словом test:

```
echo "test" > /tmp/file01.txt
```

3. Просмотрели атрибуты у только что созданного файла и разрешили чтение и запись для категории пользователей «все остальные»:

```
ls -l /tmp/file01.txt
```

```
chmod o+rw /tmp/file01.txt
```

```
ls -l /tmp/file01.txt
```

Первоначально все группы имели право на чтение, а запись могли осуществлять все, кроме «остальных пользователей».

4. От пользователя (не являющегося владельцем) попробовали прочитать файл /file01.txt:

```
cat /file01.txt
```

5. От пользователя попробовали дозаписать в файл /file01.txt слово test3 командой:

```
echo "test2" >> /file01.txt
```

6. Проверили содержимое файла командой:

```
cat /file01.txt
```

В файле теперь записано:

```
Test
```

```
Test2
```

7. От пользователя попробовали записать в файл /tmp/file01.txt слово test4, стерев при этом всю имеющуюся в файле информацию командой. Для этого воспользовалась командой `echo "test3" > /tmp/file01.txt`

8. Проверили содержимое файла командой

```
cat /tmp/file01.txt
```

9. От пользователя попробовали удалить файл /tmp/file01.txt командой `rm /tmp/file01.txt`, однако получила отказ.
10. От суперпользователя командой выполнили команду, снимающую атрибут `t` (Sticky-бит) с директории /tmp:

```
chmod -t /tmp
```

Покинули режим суперпользователя командой `exit`.

11. От пользователя проверили, что атрибута `t` у директории `/tmp` нет:

```
ls -l / | grep tmp
```

12. Повторили предыдущие шаги. Получилось удалить файл

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

14. Повысили свои права до суперпользователя и вернули атрибут `t` на директорию `/tmp` :

```
su
```

```
chmod +t /tmp
```

```
exit
```

```
guest@kzaharov:~/lab5$
guest@kzaharov:~/lab5$ cd /tmp/
guest@kzaharov:/tmp$ echo test >> file.txt
guest@kzaharov:/tmp$ chmod g+rx file.txt
guest@kzaharov:/tmp$ su guest2
Пароль:
guest2@kzaharov:/tmp$ cat file.txt
test
guest2@kzaharov:/tmp$ echo test2 >> file.txt
guest2@kzaharov:/tmp$ cat file.txt
test
test2
guest2@kzaharov:/tmp$ echo test3 > file.txt
guest2@kzaharov:/tmp$ rm file.txt
rm: невозможно удалить 'file.txt': Операция не позволена
guest2@kzaharov:/tmp$ su
Пароль:
root@kzaharov:/tmp# chmod -t /tmp
root@kzaharov:/tmp#
exit
guest2@kzaharov:/tmp$ rm file.txt
guest2@kzaharov:/tmp$
```

Figure 2.8: исследование Sticky-бита

## 3 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.

# Список литературы

1. КОМАНДА CHATTR В LINUX
2. chattr