

# **Отчёт по лабораторной работе №7**

**Основы информационной безопасности**

Бережной Иван Александрович

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>6</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
4.1	Создание программ . . . . .	8
4.2	Исследование Sticky-бита . . . . .	14
<b>5</b>	<b>Выводы</b>	<b>16</b>

## Список иллюстраций

4.1	Вход в систему под guest . . . . .	8
4.2	Создание simpleid.c 1 . . . . .	8
4.3	Создание simpleid.c 2 . . . . .	8
4.4	Компиляция simpleid.c . . . . .	9
4.5	Сравнение simpleid и команды id . . . . .	9
4.6	Модификация simpleid.c . . . . .	10
4.7	Запуск simpleid2 . . . . .	11
4.8	Смена владельца и прав . . . . .	11
4.9	Проверка атрибутов файла . . . . .	11
4.10	Запуск simpleid2 и id после смены прав . . . . .	11
4.11	Создание readfile.c 1 . . . . .	12
4.12	Создание readfile.c 2 . . . . .	12
4.13	Ограничение доступа к readfile.c . . . . .	13
4.14	Ошибка чтения readfile.c от guest . . . . .	13
4.15	Неудачные попытки чтения readfile.c . . . . .	13
4.16	Проверка Sticky-бита в /tmp . . . . .	14
4.17	Создание file01.txt в /tmp . . . . .	14
4.18	Попытка записи и удаления file01.txt от guest2 . . . . .	14
4.19	Снятие Sticky-бита с /tmp . . . . .	15
4.20	Возврат атрибута t . . . . .	15

# 1 Цель работы

Изучить механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получить практические навыки работы в консоли с дополнительными атрибутами. Рассмотреть работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

## 2 Задание

1. Создать программы
2. Исследовать Sticky-бит

## 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. [u?]

#### **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 относится 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, `gwt` — это `gw-` или `gwx`? 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 [**gcc?**].

## 4 Выполнение лабораторной работы

### 4.1 Создание программ

Войдём в систему от имени пользователя guest (рис. 4.1) и создадим программу simpleid.c (рис. 4.2 и рис. 4.3).

```
[iaberezhnoyj@iaberezhnoyj ~]$ su guest
Password:
[guest@iaberezhnoyj iaberezhnoyj]$
```

Рис. 4.1: Вход в систему под guest

```
[guest@iaberezhnoyj ~]$ touch simpleid.c
[guest@iaberezhnoyj ~]$ nano simpleid.c
```

Рис. 4.2: Создание simpleid.c 1

```
guest@iaberezhnoyj:~ — nano 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. 4.3: Создание simpleid.c 2

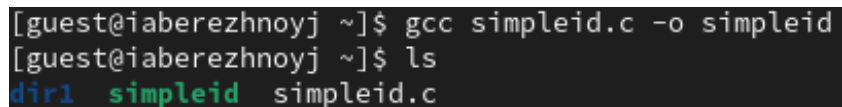


Листинг 1:

```
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

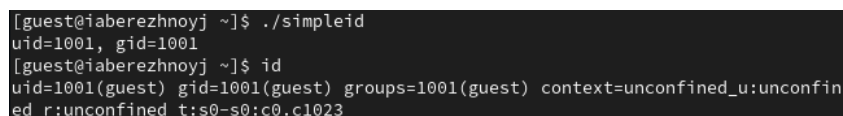
int
main ()
{
    uid_t uid = geteuid ();
    gid_t gid = getegid ();
    printf ("uid=%d, gid=%d\n", uid, gid);
    return 0;
}
```

Скомпилируем программу и убедимся, что файл создан (рис. 4.4). Выполним программу, а также команду `id`, и сравним результаты (рис. 4.5). Как видно, встроенная команда даёт больше информации.



```
[guest@iaberezhnoy ~]$ gcc simpleid.c -o simpleid
[guest@iaberezhnoy ~]$ ls
dir1 simpleid simpleid.c
```

Рис. 4.4: Компиляция `simpleid.c`



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

Рис. 4.5: Сравнение `simpleid` и команды `id`

Усовершенствуем нашу программу, добавив вывод действительных идентификаторов (рис. 4.6).

Рис. 4.6: Модификация simpleid.c

Листинг 2:

```
#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;
}
```

Скомпилируем и запустим новую программу (рис. 4.7).

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

Рис. 4.7: Запуск simpleid2

От имени суперпользователя меняем владельца файла на суперпользователя и меняем права с помощью `chmod` (рис. 4.8). Проверим правильность установки атрибутов (рис. 4.9).

```
[iaberezhnoyj@iaberezhnoyj ~]$ sudo chown root:guest /home/guest/simpleid2
[sudo] password for iaberezhnoyj:
[iaberezhnoyj@iaberezhnoyj ~]$ chmod u+s /home/guest/simpleid2
chmod: cannot access '/home/guest/simpleid2': Permission denied
[iaberezhnoyj@iaberezhnoyj ~]$ sudo chmod u+s /home/guest/simpleid2
[iaberezhnoyj@iaberezhnoyj ~]$
```

Рис. 4.8: Смена владельца и прав

```
[iaberezhnoyj@iaberezhnoyj ~]$ sudo ls -l /home/guest/simpleid2
-rwsr-xr-x. 1 root guest 17704 Apr 19 16:25 /home/guest/simpleid2
[iaberezhnoyj@iaberezhnoyj ~]$
```

Рис. 4.9: Проверка атрибутов файла

Запустим `simpleid2` и `id`. Собственная команда выводит всё ещё ограниченное количество информации (рис. 4.10).

```
[iaberezhnoyj@iaberezhnoyj ~]$ sudo /home/guest/simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[iaberezhnoyj@iaberezhnoyj ~]$ id
uid=1000(iaberezhnoyj) gid=1000(iaberezhnoyj) groups=1000(iaberezhnoyj),10(wheel)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[iaberezhnoyj@iaberezhnoyj ~]$ sudo id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 4.10: Запуск simpleid2 и id после смены прав

Создадим программу `readfile.c` (рис. 4.11 и рис. 4.12).

```
guest@iaberezhnoyj:~ — nano readfile.c
GNU nano 5.6.1 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. 4.11: Создание readfile.c 1

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

Рис. 4.12: Создание readfile.c 2

Листинг 3:

```
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
int
main (int argc, char* argv[])
```



## 4.2 Исследование Sticky-бита

Посмотрим, установлен ли атрибут Sticky на директории /tmp (рис. 4.16). В выводе присутствует буква t, значит, установлен.

```
[iaberezhnoyj@iaberezhnoyj ~]$ ls -l / | grep tmp
drwxrwxrwt. 17 root root 4096 Apr 19 16:49 tmp
```

Рис. 4.16: Проверка Sticky-бита в /tmp

От имени пользователя guest создадим файл file01.txt в директории /tmp, разрешим чтение и запись для остальных пользователей (рис. 4.17). Теперь от имени пользователя guest2 попробуем прочитать файл (успешно) и дописать что-либо в него (ошибка доступа). Также не получается удалить файл (рис. 4.18).

```
[guest@iaberezhnoyj ~]$ echo "test" > /tmp/file01.txt
[guest@iaberezhnoyj ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 Apr 19 16:50 /tmp/file01.txt
[guest@iaberezhnoyj ~]$ chmod o+rw /tmp/file01.txt
[guest@iaberezhnoyj ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 Apr 19 16:50 /tmp/file01.txt
```

Рис. 4.17: Создание file01.txt в /tmp

```
[guest@iaberezhnoyj ~]$ su guest2
Password:
[guest2@iaberezhnoyj guest]$ cat /tmp/file01.txt
test
[guest2@iaberezhnoyj guest]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Permission denied
[guest2@iaberezhnoyj guest]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': No such file or directory
```

Рис. 4.18: Попытка записи и удаления file01.txt от guest2

Повысим права до суперпользователя и снимем атрибут t с файла (рис. 4.19). Снова от имени пользователя guest2 повторим попытку выполнить команды. Кроме того, что получилось удалить файл, ничего не поменялось.

```
[iaberezhnoyj@iaberezhnoyj ~]$ su -  
Password:  
[root@iaberezhnoyj ~]# chmod -t /tmp  
[root@iaberezhnoyj ~]# su guest2  
[guest2@iaberezhnoyj root]$ ls -l / | grep tmp  
drwxrwxrwx. 18 root root 4096 Apr 19 16:53 tmp
```

Рис. 4.19: Снятие Sticky-бита с /tmp

Вернём атрибут (рис. 4.20).

```
[guest2@iaberezhnoyj root]$ su -  
Password:  
[root@iaberezhnoyj ~]# chmod +t /tmp  
[root@iaberezhnoyj ~]# exit  
logout
```

Рис. 4.20: Возврат атрибута t

## 5 Выводы

В результате выполнения работы мы изучили 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.