

Отчёта по лабораторной работе № 5

Информационная безопасность

Адебайо Ридвануллахи Айофе

Содержание

1	Цель работы	5
2	Теорическое введение	6
3	Выполнение лабораторной работы	7
3.1	Создание программы	7
3.2	Исследование Sticky-бита	13
4	Выводы	16
5	Список литературы	17

Список иллюстраций

3.1	Предварительная подготовка	7
3.2	Предварительная подготовка	8
3.3	Вход в систему и создание программы	8
3.4	Код программы simpleid.c	9
3.5	Компиляция и выполнение программы simpleid	9
3.6	Компиляция и выполнение программы simpleid2	10
3.7	Установка новых атрибутов (SetUID) и смена владельца файла . .	10
3.8	Запуск simpleid2 после установки SetGID	10
3.9	Код программы readfile.c	11
3.10	Компиляция readfile.c	11
3.11	Смена владельца и прав доступа у файла readfile.c	12
3.12	Запуск программы readfile	13
3.13	Создание файла file01.txt	14
3.14	Попытка выполнить действия над файлом file01.txt от имени пользователя guest2	14
3.15	Удаление атрибута t (Sticky-бита) и повторение действий	15
3.16	Возвращение атрибута t (Sticky-бита)	15

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 Теорическое введение

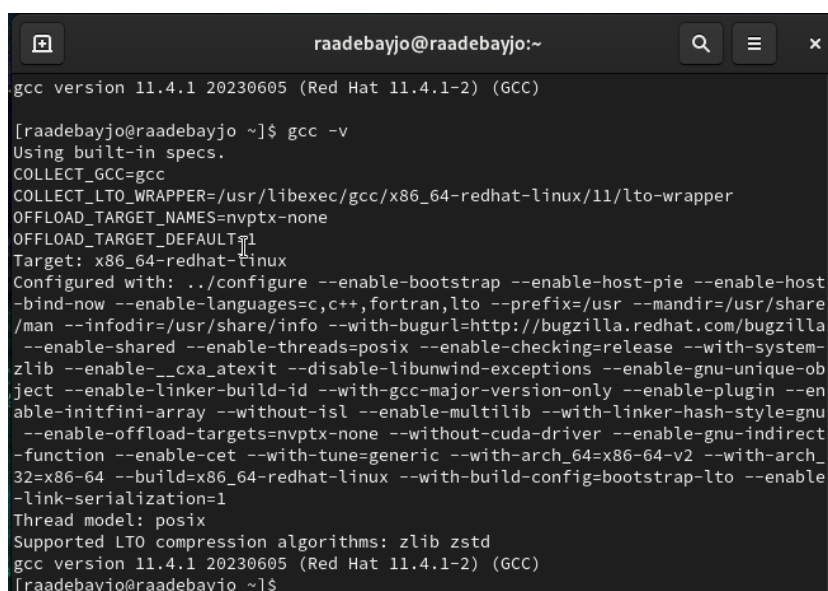
SetUID, SetGID и Sticky - это специальные типы разрешений позволяют задавать расширенные права доступа на файлы или каталоги.

- SetUID (set user ID upon execution — «установка ID пользователя во время выполнения) являются флагами прав доступа в Unix, которые разрешают пользователям запускать исполняемые файлы с правами владельца исполняемого файла.
- SetGID (set group ID upon execution — «установка ID группы во время выполнения») являются флагами прав доступа в Unix, которые разрешают пользователям запускать исполняемые файлы с правами группы исполняемого файла.
- Sticky bit в основном используется в общих каталогах, таких как /var или /tmp, поскольку пользователи могут создавать файлы, читать и выполнять их, принадлежащие другим пользователям, но не могут удалять файлы, принадлежащие другим пользователям.

3 Выполнение лабораторной работы

3.1 Создание программы

Для начала я убедился, что компилятор gcc установлен, используя команду `gcc --version` (см. 3.1).



```
raadebayjo@raadebayjo:~  
gcc version 11.4.1 20230605 (Red Hat 11.4.1-2) (GCC)  
  
[raadebayjo@raadebayjo ~]$ gcc -v  
Using built-in specs.  
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  
Target: x86_64-redhat-linux  
Configured with: ../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=http://bugzilla.redhat.com/bugzilla  
--enable-shared --enable-threads=posix --enable-checking=release --with-system-  
zlib --enable-__cxa_atexit --disable-libunwind-exceptions --enable-gnu-unique-ob  
ject --enable-linker-build-id --with-gcc-major-version-only --enable-plugin --en  
able-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  
Thread model: posix  
Supported LTO compression algorithms: zlib zstd  
gcc version 11.4.1 20230605 (Red Hat 11.4.1-2) (GCC)  
[raadebayjo@raadebayjo ~]$
```

Рис. 3.1: Предварительная подготовка

Затем отключил систему запретов до очередной перезагрузки системы командой `sudo setenforce 0`, после чего команда `getenforce` вывела `Permissive` (см. 3.2).

```
[raadebayjo@raadebayjo ~]$ setenforce 0
setenforce: security_setenforce() failed: Permission denied
[raadebayjo@raadebayjo ~]$ getenforce
Enforcing
[raadebayjo@raadebayjo ~]$ sudo setenforce 0

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for raadebayjo:
[raadebayjo@raadebayjo ~]$ getenforce
Permissive
[raadebayjo@raadebayjo ~]$
```

Рис. 3.2: Предварительная подготовка

Проверил успешное выполнение команд `whereis gcc` и `whereis g++` (см. ??)

```
[raadebayjo@raadebayjo ~]$ whereis gcc
gcc: /usr/bin/gcc /usr/lib/gcc /usr/libexec/gcc /usr/share/man/man1/gcc.1.gz /usr/share/info/gcc.info.gz
```

Вошел в систему от имени пользователя `guest` командой `su - guest`. Создал программу `simpleid.c` командой `touch simpleid.c` и открыл её в редакторе командой `vim /home/guest/simpleid.c`

```
[raadebayjo@raadebayjo ~]$ su - guest
Password:
[guest@raadebayjo ~]$ touch simpleid.c
```

Рис. 3.3: Вход в систему и создание программы

Код программы выглядит следующим образом


```

#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;
}

```

"simpleid.c" 11L, 174B

Рис. 3.4: Код программы simpleid.c

Скомпилировал программу и убедился, что файл программы был создан командой “gcc simpleid.c -o simpleid”. Выполнил программу simpleid командой “./simpleid”, а затем выполнил системную программу id командой “id”. Результаты, полученные в результате выполнения обеих команд, совпадают (uid=1001 и gid=1001)

```

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

```

Рис. 3.5: Компиляция и выполнение программы simpleid

Усложнил программу, добавив вывод действительных идентификаторов

Скомпилировал и запустил simpleid2.c командами gcc simpleid2.c -o simpleid2 и ./simpleid2

```
[guest@raadebayjo ~]$ touch simpleid2.c
[guest@raadebayjo ~]$ vim simpleid2.c
[guest@raadebayjo ~]$ gcc simpleid1.c -o simpleid2
cc1: fatal error: simpleid1.c: No such file or directory
compilation terminated.
[guest@raadebayjo ~]$ gcc simpleid2.c -o simpleid2
[guest@raadebayjo ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@raadebayjo ~]$
```

Рис. 3.6: Компиляция и выполнение программы simpleid2

От имени суперпользователя выполнил команды `sudo chown root:guest /home/guest/simpleid2` и `sudo chmod u+s /home/guest/simpleid2`, затем выполнил проверку правильности установки новых атрибутов и смены владельца файла `simpleid2` командой `sudo ls -l /home/guest/simpleid2`. Этими командами была произведена смена пользователя файла на `root` и установлен SetUID-бит.

```
[guest@raadebayjo ~]$ su
Password:
[root@raadebayjo guest]# chown root:guest /home/guest/simpleid2
[root@raadebayjo guest]# chmod u+s /home/guest/simpleid2
[root@raadebayjo guest]# ls -l simpleid2
-rwsr-xr-x. 1 root guest 24488 Oct  3 20:34 simpleid2
[root@raadebayjo guest]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@raadebayjo guest]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@raadebayjo guest]#
```

Рис. 3.7: Установка новых атрибутов (SetUID) и смена владельца файла

Запустил программы `simpleid2` и `id`. Теперь появились различия в `uid`

```
[guest@raadebayjo ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@raadebayjo ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@raadebayjo ~]$
```

Рис. 3.8: Запуск simpleid2 после установки SetGID

Создаем программу readfile.c

```
#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;
}
```

"readfile.c" 24L, 406B 23,1

Рис. 3.9: Код программы readfile.c

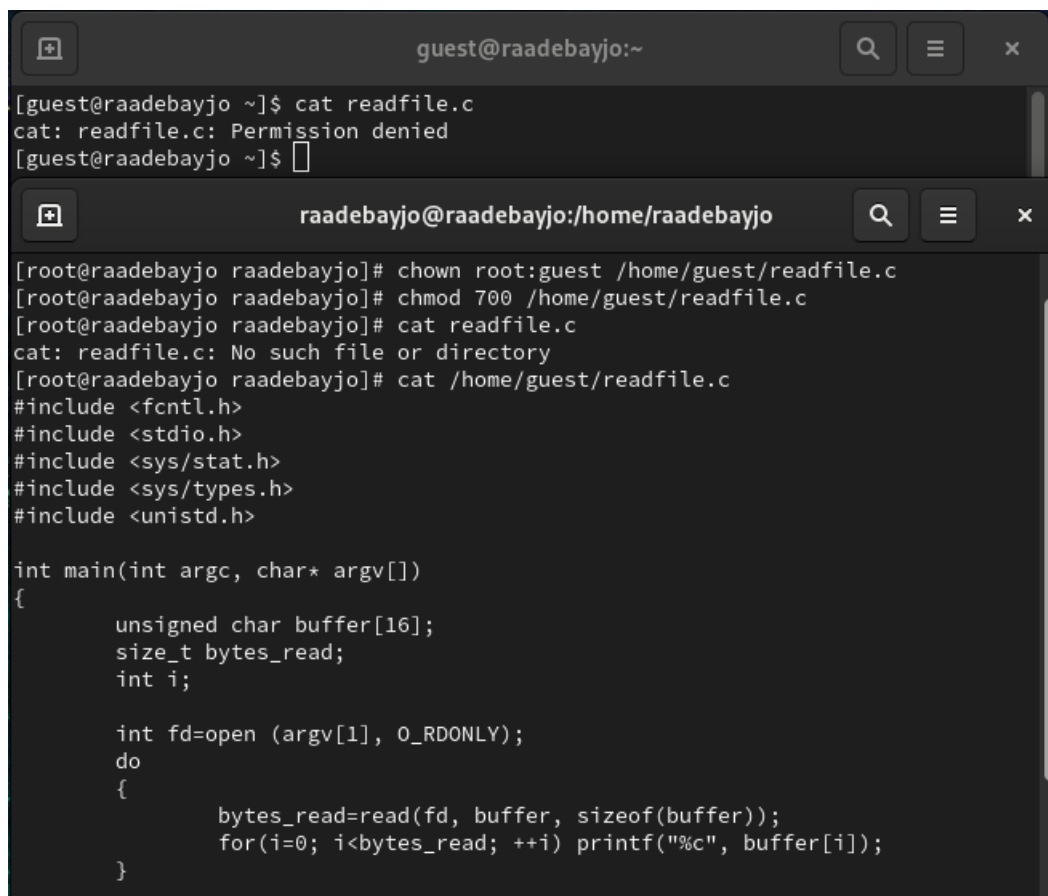
Скомпилировал созданную программу командой `gcc readfile.c -o readfile`.

```
[guest@raadebayjo ~]$ touch readfile.c
[guest@raadebayjo ~]$ vim readfile.c
[guest@raadebayjo ~]$ vim readfile.c
[guest@raadebayjo ~]$ gcc readfile.c -o readfile
readfile.c: In function 'main':
readfile.c:17:66: error: expected ']' before '(' token
   17 |         for(i=0; i<bytes_read; ++i) printf("%c", buffer[i]);
      |                                     ^
[guest@raadebayjo ~]$ vim readfile.c
[guest@raadebayjo ~]$ gcc readfile.c -o readfile
[guest@raadebayjo ~]$
```

Рис. 3.10: Компиляция readfile.c

Сменил владельца у файла readfile.c командой `sudo chown root:guest /home/guest/readfile.c` и поменял права так, чтобы только суперпользователь мог прочитать его, а guest не мог, с помощью команды `sudo chmod 700`

/home/guest/readfile.c. Теперь убедился, что пользователь guest не может прочитать файл readfile.c командой `cat readfile.c`, получив отказ в доступе



```
guest@raadebayjo:~  
[guest@raadebayjo ~]$ cat readfile.c  
cat: readfile.c: Permission denied  
[guest@raadebayjo ~]$  
  
raadebayjo@raadebayjo:/home/raadebayjo  
[root@raadebayjo raadebayjo]# chown root:guest /home/guest/readfile.c  
[root@raadebayjo raadebayjo]# chmod 700 /home/guest/readfile.c  
[root@raadebayjo raadebayjo]# cat readfile.c  
cat: readfile.c: No such file or directory  
[root@raadebayjo raadebayjo]# 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]);  
    }
```

Рис. 3.11: Смена владельца и прав доступа у файла readfile.c

Поменял владельца у программы readfile и установила SetUID. Проверил, может ли программа readfile прочитать файл readfile.c командой `./readfile readfile.c`. **Прочитать удалось**. Аналогично проверил, можно ли прочитать файл /etc/shadow. **Прочитать удалось**

```
[root@raadebayjo raadebayjo]# chown root:guest /home/guest/readfile
[root@raadebayjo raadebayjo]# chmod u+s /home/guest/readfile
[root@raadebayjo raadebayjo]# █

guest@raadebayjo:~
cat: readfile.c: Permission denied
[guest@raadebayjo ~]$ ./readfile 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);
}
```

Рис. 3.12: Запуск программы readfile

3.2 Исследование Sticky-бита

Командой `ls -l / | grep tmp` убеждался, что атрибут Sticky на директории `/tmp` установлен. От имени пользователя `guest` создал файл `file01.txt` в директории `/tmp` со словом `test` командой `echo "test" > /tmp/file01.txt`. Просмотрел атрибуты у только что созданного файла и разрешаем чтение и запись для категории пользователей *все остальные* командами `ls -l /tmp/file01.txt` и `chmod o+rw /tmp/file01.txt`.

```
[raadebayjo@raadebayjo ~]$ ls -l / | grep tmp
drwxrwxrwt. 17 root root 4096 Oct  3 21:04 tmp

[guest@raadebayjo ~]$ echo "test" > /tmp/file01.txt
[guest@raadebayjo ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 Oct  3 21:10 /tmp/file01.txt
[guest@raadebayjo ~]$ chmod o+rw /tmp/file01.txt
[guest@raadebayjo ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 Oct  3 21:10 /tmp/file01.txt
[guest@raadebayjo ~]$
```

Рис. 3.13: Создание файла file01.txt

От имени пользователя guest2 попробовал прочитать файл командой `cat /tmp/file01.txt` - это удалось. Далее попытался дозаписать в файл слово `test2`, проверить содержимое файла и записать в файл слово `test3`, стерев при этом всю имеющуюся в файле информацию - эти операции удалось выполнить только в случае, если еще дополнительно разрешить чтение и запись для группы пользователей командой `chmod g+rw /tmp/file01.txt`. От имени пользователя guest2 попробовал удалить файл - это не удастся ни в каком из случаев, возникает ошибка.

```
[raadebayjo@raadebayjo ~]$ su - guest2
Password:
[guest2@raadebayjo ~]$ cat /tmp/file01.txt
test
[guest2@raadebayjo ~]$ echo "test2" > /tmp/file01.txt
-bash: /tmp/file01.txt: Permission denied
[guest2@raadebayjo ~]$ cat /tmp/file01.txt
test
[guest2@raadebayjo ~]$ echo "test3" > /tmp/file01.txt
[guest2@raadebayjo ~]$ cat /tmp/file01.txt
test3
[guest2@raadebayjo ~]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest2@raadebayjo ~]$
```

```
guest@raadebayjo:~
-rw-r--rw-. 1 guest guest 5 Oct  3 21:10 /tmp/file01.txt
[guest@raadebayjo ~]$ chmod g+rw /tmp/file01.txt
[guest@raadebayjo ~]$
```

Рис. 3.14: Попытка выполнить действия над файлом file01.txt от имени пользователя guest2

Повысила права до суперпользователя командой `su -` и выполнила команду, снимающую атрибут `t` с директории `/tmp` командой `chmod -t /tmp`. После чего покинула режим суперпользователя командой `exit`. Повторила предыдущие шаги. Теперь

мне удалось удалить файл file01.txt от имени пользователя, не являющегося его владельцем

```
[guest2@raadebayjo ~]$ ls -l / | grep tmp
drwxrwxrwx. 17 root root 4096 Oct  3 21:21 tmp
[guest2@raadebayjo ~]$ cat /tmp/file01.txt
test3
[guest2@raadebayjo ~]$ echo "test2" > /tmp/file01.txt
[guest2@raadebayjo ~]$ rm /tmp/file01.txt
[guest2@raadebayjo ~]$
```

root@raadebayjo:~

```
drwxrwxrwt. 17 root root 4096 Oct  3 21:04 tmp
[raadebayjo@raadebayjo ~]$ su -
Password:
[root@raadebayjo ~]# chmod -t /tmp
[root@raadebayjo ~]#
```

Рис. 3.15: Удаление атрибута t (Sticky-бита) и повторение действий

Повысила свои права до суперпользователя и вернула атрибут t на директорию /tmp

```
[root@raadebayjo ~]# chmod +t /tmp
[root@raadebayjo ~]# exit
logout
[raadebayjo@raadebayjo ~]$
```

Рис. 3.16: Возвращение атрибута t (Sticky-бита)

4 Выводы

В ходе выполнения данной лабораторной работы я изучил механизмы изменения идентификаторов, применение SetUID- и Sticky-битов. Получил практические навыки работы в консоли с дополнительными атрибутами. Рассмотрел работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

5 Список литературы

1. Кулябов Д. С. *Лабораторная работа №5**: 005-lab_discret_sticky.pdf*
2. Использование SETUID, SETGID и Sticky bit для расширенной настройки прав доступа в операционных системах Linux [Электронный ресурс]. 2023.URL: <https://ruvds.com/ru/helpcenter/suid-sgid-sticky-bit-linux/> (дата обращения: 05.10.2023)