

Лабораторная работа 5

Терентьев Егор Дмитриевич, НФИбд-01-19

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ПРЕЗЕНТАЦИЯ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

дисциплина: Информационная безопасность

Преподаватель: Кулябов Дмитрий Сергеевич

Студент: Терентьев Егор Дмитриевич

Группа: НФИбд-01-19

МОСКВА

2022 г.

Прагматика выполнения лабораторной работы

Прагматика выполнения лабораторной работы

- ▶ Изучение механизмов изменения идентификаторов:
 1. Применения SetUID и SetGID
 2. Применения Sticky-битов.

Цель работы

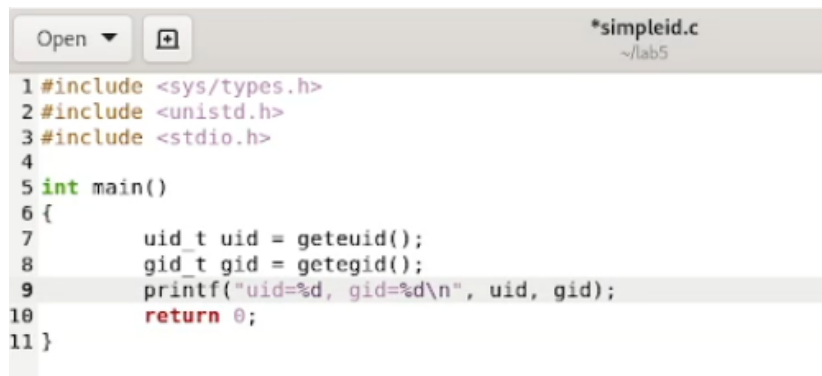
Цель работы

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.

Выполнение лабораторной работы

1. Создали программу simpleid

1. Создали программу simpleid



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

Figure 1: simpleid

2. Сравниваем `id` и `simpleid`

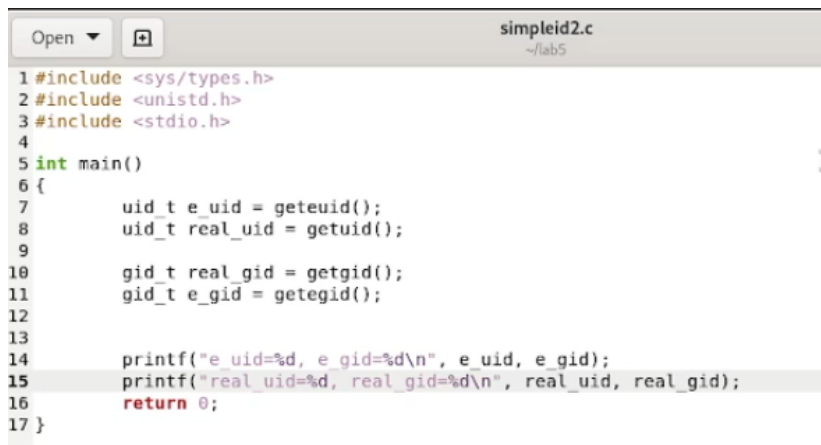
2. Сравниваем id и simpleid

```
[guest@edterentjev lab5]$ ./simpleid
uid=1002, gid=1002
[guest@edterentjev lab5]$ id
uid=1002(guest) gid=1002(guest) groups=1002(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Figure 2: id_simpleid

3. Создаем simpleid2

3. Создаем simpleid2



```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4
5 int main()
6 {
7     uid_t e_uid = geteuid();
8     uid_t real_uid = getuid();
9
10    gid_t real_gid = getgid();
11    gid_t e_gid = getegid();
12
13
14    printf("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
15    printf("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
16    return 0;
17 }
```

Figure 3: simpleid2

4. Запускаем simpleid2

4. Запускаем simpleid2

```
[guest@edterentjev lab5]$ gedit simpleid2.c  
[guest@edterentjev lab5]$ gcc simpleid2.c -o simpleid2  
[guest@edterentjev lab5]$ ./simpleid2  
e_uid=1002, e_gid=1002  
real_uid=1002, real_gid=1002
```

Figure 4: chown_chmod

5. Меняем chown и chmod

5. Меняем chown и chmod

```
[guest@edterentjev lab5]$ su
Password:
[root@edterentjev lab5]# chown root:guest /home/guest/simpleid2
chown: cannot access '/home/guest/simpleid2': No such file or directory
[root@edterentjev lab5]# chown root:guest /home/guest/lab5/simpleid2
[root@edterentjev lab5]# chmod u+s /home/guest/lab5/simpleid2
```

Figure 5: chown_chmod_simpleid2

6. Сравниваем `id` и `simpleid2`

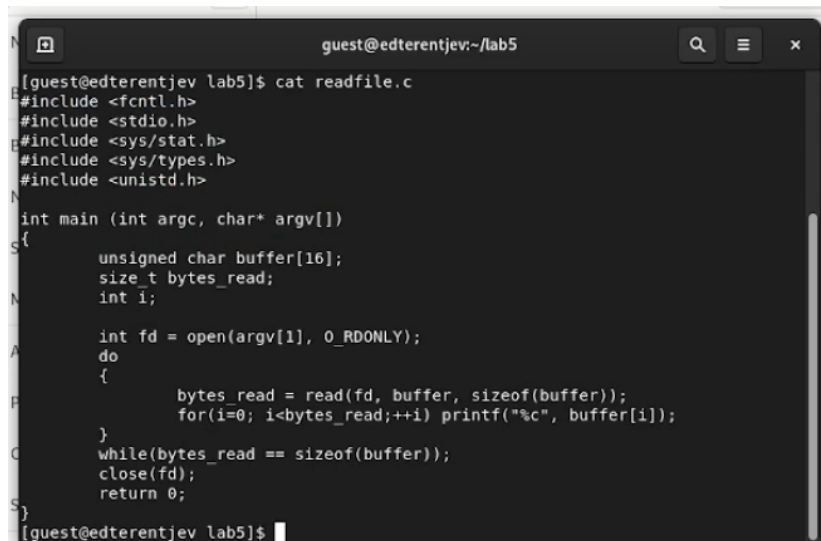
6. Сравниваем id и simpleid2

```
[root@edterentjev lab5]# ls -l simpleid2
-rwsrwxr-x. 1 root guest 26008 Oct  4 15:20 simpleid2
[root@edterentjev lab5]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@edterentjev lab5]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfi
ned t:s0-s0:c0.c1023
```

Figure 6: id_simpleid2

7. Создаем readfile

7. Создаем readfile

A terminal window with a dark background and light text. The title bar at the top reads "guest@edterentjev: ~/lab5". On the left side of the terminal, there are vertical labels: "N", "E", "E", "N", "S", "A", "P", "C", "S". The terminal shows the command "cat readfile.c" and its output, which is the C code for a file reader. The code includes headers for file control, standard I/O, system statistics, system types, and UNIX standards. The main function opens a file in read-only mode and reads it into a 16-byte buffer, printing each byte as a character. The code ends with a return statement and a closing shell prompt.

```
guest@edterentjev: ~/lab5
[guest@edterentjev lab5]$ 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;

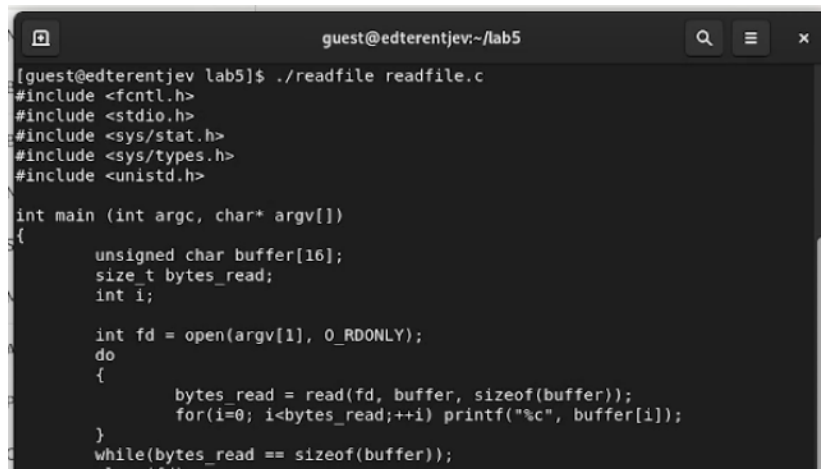
    int fd = open(argv[1], O_RDONLY);
    do
    {
        bytes_read = read(fd, buffer, sizeof(buffer));
        for(i=0; i<bytes_read;++i) printf("%c", buffer[i]);
    }
    while(bytes_read == sizeof(buffer));
    close(fd);
    return 0;
}
[guest@edterentjev lab5]$
```

Figure 7: readfile

8. Проверяем перезапись, удаление, переименование

8. Проверяем перезапись, удаление, переименование

Затем меняем владельца файла чтобы только суперпользователь мог прочитать его и проверяем может ли пользователь прочитать readfile с помощью readfile.o

A terminal window with a dark background. The title bar shows the user 'guest' at host 'edterentjev' in directory '~/lab5'. The terminal displays the command to run a program and the source code of that program. The code is a C program that opens a file in read-only mode and prints its contents in chunks of 16 bytes.

```
guest@edterentjev:~/lab5
[guest@edterentjev lab5]$ ./readfile readfile.c
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

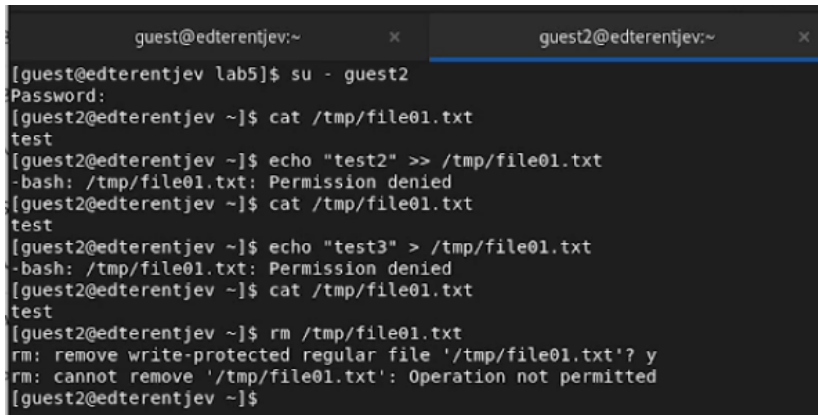
int main (int argc, char* argv[])
{
    unsigned char buffer[16];
    size_t bytes_read;
    int i;

    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));
}
```

9. Создаем файл test1 от guest и работаем
с guest2

9. Создаем файл test1 от guest и работаем с guest2

От пользователя guest2 (не являющегося владельцем) попробуйте прочитать файл /tmp/file01.txt От пользователя guest2 попробуйте дозаписать в файл /tmp/file01.txt слово test2 Проверьте содержимое файла От пользователя guest2 попробуйте записать в файл /tmp/file01.txt Проверьте содержимое файла От пользователя guest2 попробуйте удалить файл /tmp/file01.txt



```
guest@edterentjev:~ x guest2@edterentjev:~ x
[guest@edterentjev lab5]$ su - guest2
Password:
[guest2@edterentjev ~]$ cat /tmp/file01.txt
test
[guest2@edterentjev ~]$ echo "test2" >> /tmp/file01.txt
-bash: /tmp/file01.txt: Permission denied
[guest2@edterentjev ~]$ cat /tmp/file01.txt
test
[guest2@edterentjev ~]$ echo "test3" > /tmp/file01.txt
-bash: /tmp/file01.txt: Permission denied
[guest2@edterentjev ~]$ cat /tmp/file01.txt
test
[guest2@edterentjev ~]$ rm /tmp/file01.txt
rm: remove write-protected regular file '/tmp/file01.txt'? y
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest2@edterentjev ~]$
```

Figure 9: file1_guest2

10. Снимаем sticky-bit

10. Снимаем sticky-bit

```
[guest2@edterentjev ~]$ su -  
Password:  
[root@edterentjev ~]# chmod -t /tmp  
[root@edterentjev ~]# su - guest2  
[guest2@edterentjev ~]$ ls -l / | grep tmp  
drwxrwxrwx. 17 root root 4096 Oct  4 15:59 tmp  
[guest2@edterentjev ~]$ echo "test2" >> /tmp/file01.txt  
-bash: /tmp/file01.txt: Permission denied  
[guest2@edterentjev ~]$ echo "test2" > /tmp/file01.txt  
-bash: /tmp/file01.txt: Permission denied  
[guest2@edterentjev ~]$ rm /tmp/file01.txt  
rm: remove write-protected regular file '/tmp/file01.txt'? y
```

Figure 10: guest2_without_stickybit

Выводы

Выводы

В результате выполнения работы я изучил механизмы изменения идентификаторов, применения SetUID- и Sticky-битов, получил практические навыки работы в консоли с дополнительными атрибутами, а также рассмотрел работы механизма смены идентификатора процессов пользователей и влияние бита Sticky на запись и удаление файлов.

