Отчёт по лабораторной работе 5

Макухина Анастасия Вадимовна

Содержание

# 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. Войдём в систему от имени пользователя guest

[Рисунок 1](Images/1.png).

1. Создайте программу simpleid.c:

nano 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”, uid, gid);

return 0;

}

[Рисунок 2](Images/2.png), [Рисунок 3](Images/3.png).

1. Скомплилируем программу и убедимся, что файл программы создан:

gcc simpleid.c -o simpleid - [Рисунок 4](Images/4.png).

1. Выполним программу simpleid:

./simpleid - [Рисунок 5](Images/5.png).

1. Выполните системную программу id:

id - [Рисунок 6](Images/6.png).

Результаты одинаковы.

1. Усложним программу, добавив вывод действительных идентификаторов:

#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”, e\_uid, e\_gid);

printf (“real\_uid=%d, real\_gid=%d”, real\_uid,

,→ real\_gid);

return 0;

}

Получившуюся программу назовём simpleid2.c - [Рисунок 7](Images/7.png).

1. Скомпилируем и запустим simpleid2.c:

gcc simpleid2.c -o simpleid2

./simpleid2 - [Рисунок 8](Images/8.png).

1. От имени суперпользователя выполним команды:

chown root:guest /home/guest/simpleid2

chmod u+s /home/guest/simpleid2 - [Рисунок 9](Images/9.png).

1. Повысим временно свои права с помощью su.

su - [Рисунок 10](Images/10.png).

1. Выполним проверку правильности установки новых атрибутов и смены владельца файла simpleid2:

ls -l simpleid2 - [Рисунок 11](Images/11.png).

1. Запустим simpleid2 и id:

./simpleid2

id - [Рисунок 12](Images/12.png).

Результаты одинаковы.

1. Проделайте тоже самое относительно SetGID-бита - [Рисунок 13](Images/13.png).
2. Создадим программу 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;

}

[Рисунок 14](Images/14.png).

1. Откомпилируем её

gcc readfile.c -o readfile - [Рисунок 15](Images/15.png).

1. Сменим владельца у файла readfile.c (или любого другого текстового файла в системе) и изменим права так, чтобы только суперпользователь (root) мог прочитать его, a guest не мог.

[Рисунок 16](Images/16.png).

1. Проверим, что пользователь guest не может прочитать файл readfile.c

[Рисунок 17](Images/17.png).

1. Сменим у программы readfile владельца и установим SetU’D-бит

[Рисунок 18](Images/18.png).

1. Проверим, может ли программа readfile прочитать файл readfile.c

[Рисунок 19](Images/19.png), [Рисунок 20](Images/20.png).

1. Проверим, может ли программа readfile прочитать файл /etc/shadow

[Рисунок 21](Images/21.png).

1. Выясним, установлен ли атрибут Sticky на директории /tmp, для чего выполним команду:

ls -l / | grep tmp - [Рисунок 22](Images/22.png).

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

echo “test” > /tmp/file01.txt - [Рисунок 23](Images/23.png).

1. Просмотрим атрибуты у только что созданного файла и разрешим чтение и запись для категории пользователей «все остальные»:

ls -l /tmp/file01.txt

chmod o+rw /tmp/file01.txt

ls -l /tmp/file01.txt - [Рисунок 24](Images/24.png).

1. От пользователя guest2 (не являющегося владельцем) попробуем прочитать файл /tmp/file01.txt:

cat /tmp/file01.txt - [Рисунок 25](Images/25.png).

1. От пользователя guest2 попробуем дозаписать в файл /tmp/file01.txt слово test2 командой:

echo “test2” > /tmp/file01.txt - [Рисунок 26](Images/26.png).

Да, удалось.

1. Проверим содержимое файла командой:

cat /tmp/file01.txt - [Рисунок 27](Images/27.png).

1. От пользователя guest2 попробуем записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой:

echo “test3” > /tmp/file01.txt - [Рисунок 28](Images/28.png).

1. Проверьте содержимое файла командой:

cat /tmp/file01.txt - [Рисунок 29](Images/29.png).

1. От пользователя guest2 попробуем удалить файл /tmp/file01.txt командой:

rm /tmp/fileOl.txt - [Рисунок 30](Images/30.png).

Не удалось.

1. Повысим свои права до суперпользователя следующей командой

su

и выполним после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp:

chmod -t /tmp - [Рисунок 31](Images/31.png).

1. Покинем режим суперпользователя командой

exit - [Рисунок 32](Images/32.png).

1. От пользователя guest2 проверим, что атрибута t у директории /tmp нет:

ls -l / | grep tmp - [Рисунок 33](Images/33.png).

1. Повторите предыдущие шаги - [Рисунок 34](Images/34.png).

Изменений не произошло.

1. Проверим, можно ли удалить файл от имени пользователя, не являющегося его владельцем - [Рисунок 35](Images/35.png).

Удалось.

1. Повысим свои права до суперпользователя и вернём атрибут t на директорию /tmp:

su -

chmod +t /tmp

exit - [Рисунок 36](Images/36.png).

# Выводы

В ходе выяполнения работы я изученла механизмы изменения идентификаторов, применения SetUID- и Sticky-битов, полученила практические навыки работы в консоли с дополнительными атрибутами, рассмотрела работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.