Отчет по лабораторной работе №5

Дисциплина: Информационная безопасность

Выполнила: Афтаева Ксения Васильевна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучение механизмов изменения, 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 Задачи

1. Подготовить лабораторный стенд, если это требуется.
2. Выполнить задания по созданию и компилированию программ.
3. Выполнить задания по исследованию Sticky-бита.

3 Теоретическое введение

Изначально каждый файл имеет три параметра доступа [1]:

- **чтение** - разрешает прочитать содержимое файла или каталога (r);
- **запись** - разрешает записывать новые данные в файл или изменять существующие, а также позволяет создавать и изменять файлы и каталоги (w);
- **выполнение** - разрешает выполнять, как программу, и входить в директорию (x).

Каждый файл имеет три категории пользователей, для которых можно устанавливать различные сочетания прав доступа:

- **владелец** - набор прав для владельца файла, пользователя, который его создал или сейчас установлен его владельцем;
- **группа** - любая группа пользователей, существующая в системе и привязанная к файлу;
- **остальные** - все пользователи, кроме владельца и пользователей, входящих в группу файла.

Информация о правах доступа к файлу представлена в виде **10** символов. Первый символ определяет тип файла. Если первый символ -, то это обычный файл. Если первый символ d, то это каталог. Следующие 3 символа показывают разрешения для владельца. Буква означает наличие разрешения, а прочерк — его

отсутствие. Следующие 3 символа показывают разрешения для группы. Порядок записи разрешений всегда такой: чтение, запись, выполнение. Последние 3 символа показывают разрешения для всех остальных пользователей[2].

Для того, чтобы позволить обычным пользователям выполнять программы от имени суперпользователя без знания его пароля придуманы SetUID и SetGID биты [3].

- **SetUID** - если этот бит установлен, то при выполнении программы, id пользователя, от которого она запущена заменяется на id владельца файла. Фактически, это позволяет обычным пользователям запускать программы от имени суперпользователя;
- **SetGID** - этот флаг работает аналогичным образом, только разница в том, что пользователь считается членом группы, с которой связан файл, а не групп, к которым он действительно принадлежит. Если SGID флаг установлен на каталог, все файлы, созданные в нем, будут связаны с группой каталога, а не пользователя. Такое поведение используется для организации общих папок;
- **Sticky-bit** - этот бит тоже используется для создания общих папок. Если он установлен, то пользователи могут только создавать, читать и выполнять файлы, но не могут удалять файлы, принадлежащие другим пользователям.

Для изменения владельца и группы файла или каталога есть команда chown [4].

Используется следующий шаблон выполнения данной команды chown:

```
sudo chown имя_нового_владельца:имя_новой_группы имя_файла_или_директории
```

4 Выполнение лабораторной работы

4.1 Создание программы

1. Вошла в систему от имени пользователя guest (рис. 4.1). Убедилась, что в системе установлен компилятор gcc, проверив версию командой `gcc -v` (рис. 4.1). Видим, что он установлен. Отключила систему запретов до очередной перезагрузки командой `setenforce 0` (рис. 4.1). После этого команда `getenforce` выводит “Permissive” (рис. 4.1).

Рис. 4.1: Подготовка лабораторного стенда

2. Создала программу simpleid.c в текстовом редакторе nano (рис. 4.2).

3. Скомпилировала программу и убедилась, что файл программы создан, командой `gcc simpleid.c -o simpleid` (рис. 4.2).

4. Выполнила программу simpleid, введя `./simpleid` (рис. 4.2).

5. Выполнила системную программу id (рис. 4.2). Видим, что выводимая информация о uid и gid идентична, но команда `id` также выводит groups=1001(guest).

```
[guest@kvaftaeva ~]$ nano simpleid.c
[guest@kvaftaeva ~]$ cat 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;
}
[guest@kvaftaeva ~]$ gcc simpleid.c -o simpleid
[guest@kvaftaeva ~]$ ./simpleid
uid=1001, gid=1001
[guest@kvaftaeva ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unc
```

Рис. 4.2: Программа simpleid

6. Усложнила программу, добавив вывод действительных идентификаторов (рис. 4.3). Получившуюся программу назвала simpleid2.c (рис. 4.3).

7. Скомпилировала программу командой `gcc simpleid2.c -o simpleid2` (рис. 4.3). Запустила программу, введя `./simpleid2` (рис. 4.3).

```
[guest@kvaftaeva ~]$ nano simpleid.c
[guest@kvaftaeva ~]$ cat simpleid2.c
#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;
}
[guest@kvaftaeva ~]$ gcc simpleid2.c -o simpleid2
[guest@kvaftaeva ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
```

Рис. 4.3: Программа simpleid2

8. Выполнила команды `sudo chown root:guest /home/guest/simpleid2` и `sudo chmod u+s /home/guest/simpleid2` (рис. 4.4).

9. Первая команда меняет у файла владельца и группу. Вторая команда ставит SetUID-бит.

10. Выполнила проверку правильности установки новых атрибутов и смены владельца файла simpleid2 с помощью команды `ls -l simpleid2` (рис. 4.4). Видим, что все установлено верно.

11. Запустила файл simpleid2 командой `./simpleid2` (рис. 4.4). Также выполнила системную команду `id` (рис. 4.4). Видим, что файл выводит информацию не только о реальном идентификаторе, но и эффективных ID - EUID и EGID. Фактические ID пользователя и группы соответствуютт ID пользователя, который вызвал процесс (пользователь guest, группа guest). Эффективный ID пользователя соответствует установленному SetUid биту на исполняемом файле (привелегии суперпользователя). Эффективный ID группы соответствует установленному SetGid биту на исполняемом файле.Команда `id` выводит информацию только о реальном идентификаторе.

12. Проделала то же самое относительно SetGID-бита (рис. 4.4). Видим, что теперь при исполнении файла выводится EUID = 1001, EGID = 0, так как мы установили SetGid бит.

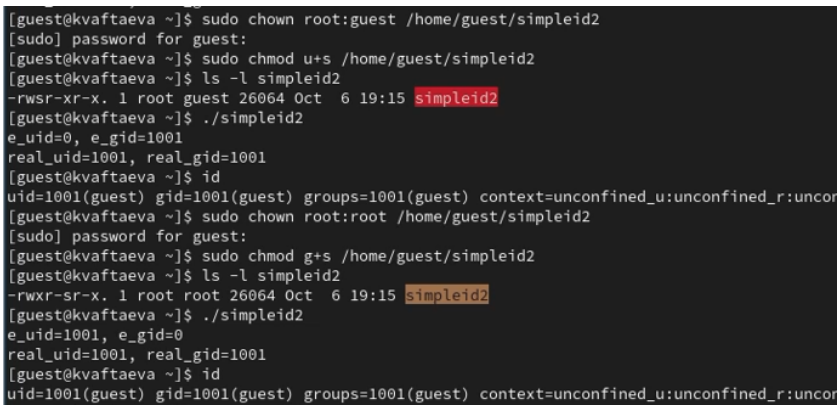

Рис. 4.4: SetUID- и SetGID-биты

13. Создала программу readfile с помощью тектового редактора nano (рис. 4.5).

14. Откомпилировала её командой `gcc readfile.c -o readfile` (рис. 4.5).

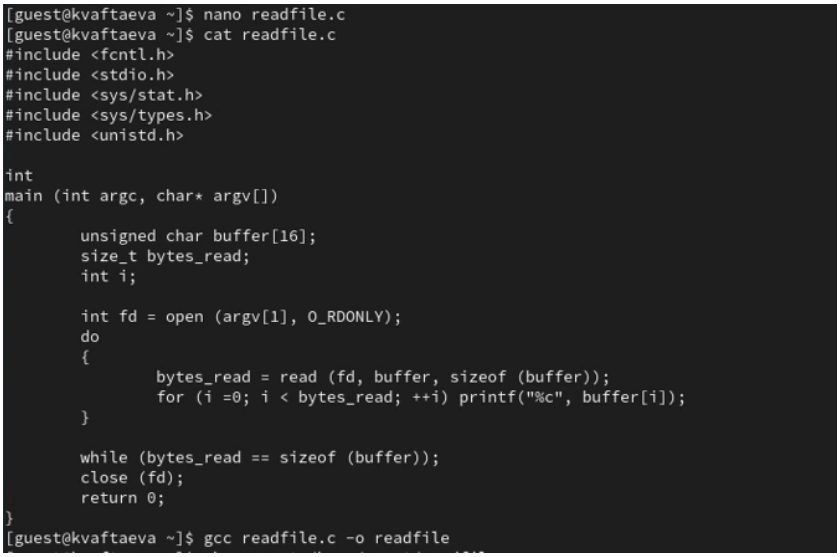

Рис. 4.5: Файл readfile.c

15. Сменила владельца у файла readfile.c на root командой `sudo chown root /home/guest/readfile` и изменила права так, чтобы только суперпользователь (в моем случае владелец) мог прочитать его, а guest (член группы) не мог командой `sudo chmod 733 /home/guest/readfile` (рис. 4.6).

16. Проверила, что пользователь guest не может прочитать файл readfile.c командой `cat readfile.c` (рис. 4.6). Видим, что нам действильно отказано в доступе.

17. Сменила у программы readfile владельца обратно на guest и установила SetUID-бит командами `sudo chown guest:guest /home/guest/readfile` и `sudo chmod u+s /home/guest/readfile` (рис. 4.6).

```
[guest@kvaftaeva ~]$ chown root /home/guest/readfile
chown: changing ownership of '/home/guest/readfile': Operation not permitted
[guest@kvaftaeva ~]$ sudo chown root /home/guest/readfile
[sudo] password for guest:
[guest@kvaftaeva ~]$ chmod 733 /home/guest/readfile
chmod: changing permissions of '/home/guest/readfile': Operation not permitted
[guest@kvaftaeva ~]$ sudo chmod 733 /home/guest/readfile
[guest@kvaftaeva ~]$ ls -l readfile
-rwx-wx-wx. 1 root guest 26008 Oct  6 19:42 readfile
[guest@kvaftaeva ~]$ cat readfile
cat: readfile: Permission denied
[guest@kvaftaeva ~]$ sudo chown guest:guest /home/guest/readfile
[guest@kvaftaeva ~]$ sudo chmod u+s /home/guest/readfile
```

Рис. 4.6: Смена владельца файла readfile.c

18. Проверила, может ли программа readfile прочитать файл readfile.c (рис. 4.7). Видим, что может, так как мы установили SetUID-бит.

```
[guest@kvaftaeva ~]$ ./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]);
        }
                                I
        while (bytes_read == sizeof (buffer));
        close (fd);
        return 0;
}
[guest@kvaftaeva ~]$
```

Рис. 4.7: Чтение файла readfile.c

19. Проверилп, может ли программа readfile прочитать файл /etc/shadow (рис. 4.8). Видим, что может, так как мы установили SetUID-бит.

Рис. 4.8: Чтение файла /etc/shadow

4.2 Исследование Sticky-бита

1. Выяснила, установлен ли атрибут Sticky на директории /tmp, для чего выполнила команду `ls -l / | grep tmp` (рис. 4.9). Видим, что он установлен (буква t в конце).

2. От имени пользователя guest создала файл file01.txt в директории /tmp со словом test командой `echo "test" > /tmp/file01.txt` (рис. 4.9).

3. Просмотрела атрибуты у только что созданного файла (`команда ls -l /tmp/file01.txt`) и разрешила чтение и запись для категории пользователей «все остальные» командой `chmod o+rw /tmp/file01.txt` (рис. 4.9). Проверила атрибуты (`команда ls -l /tmp/file01.txt`) (рис. 4.9).

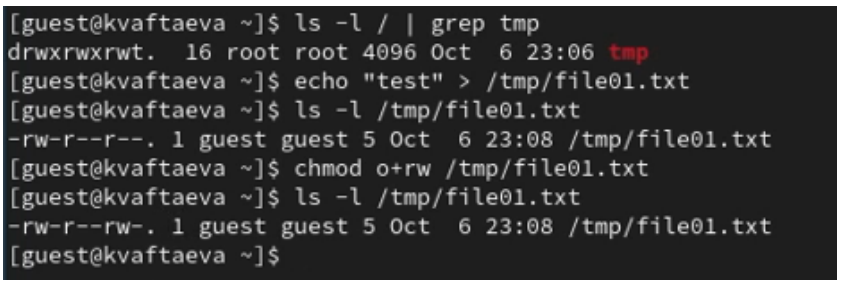

Рис. 4.9: Проверка наличия Sticky-бита на директории /tmp

4. От пользователя guest2 (не являющегося владельцем) попробовала прочитать файл /tmp/file01.txt командой `cat /tmp/file01.txt` (рис. 4.10). Прочитать файл удалось.

5. От пользователя guest2 попробовала дозаписать в файл /tmp/file01.txt слово test2 командой `echo "test2" >> /tmp/file01.txt` (рис. 4.10). Выполнить данную операцию не удалось.

6. Проверила содержимое файла командой `cat /tmp/file01.txt` (рис. 4.10). Действие по сути бессмысленное, так как файл не менялся.

7. От пользователя guest2 попробовала записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой `echo "test3" > /tmp/file01.txt` (рис. 4.10). Выполнить эту операцию не удалось.

8. Проверила содержимое файла командой `cat /tmp/file01.txt` (рис. 4.10). Действие по сути бессмысленное, так как файл не менялся.

9. От пользователя guest2 попробовала удалить файл /tmp/file01.txt командой `rm /tmp/fileOl.txt` (рис. 4.10).

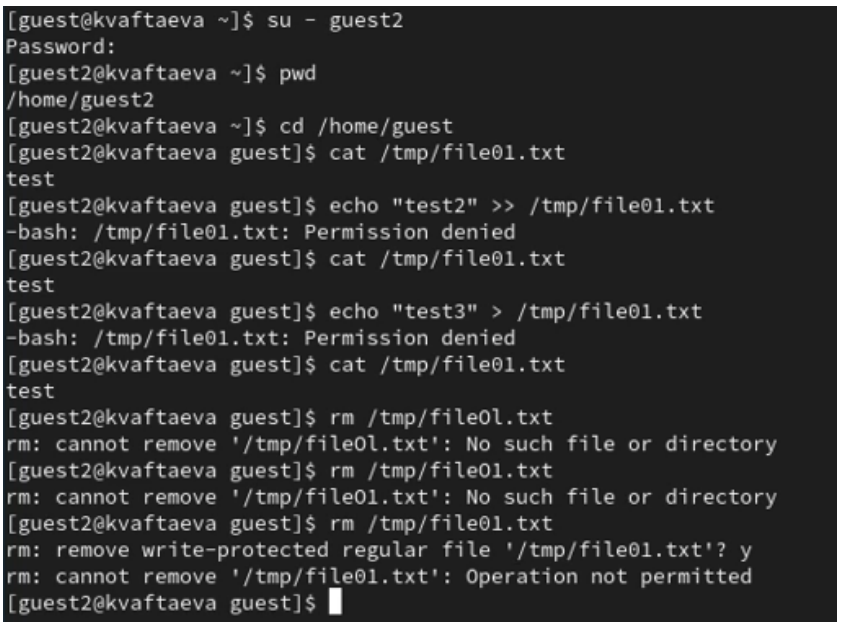

Рис. 4.10: Проверка возможности действий при наличии Sticky-бита

10. Повысила свои права до суперпользователя следующей командой `su -` и выполнида после этого команду `chmod -t /tmp`, снимающую атрибут t (Sticky-бит) с директории /tmp (рис. 4.11).

11. Покинула режим суперпользователя командой `exit` (рис. 4.11)

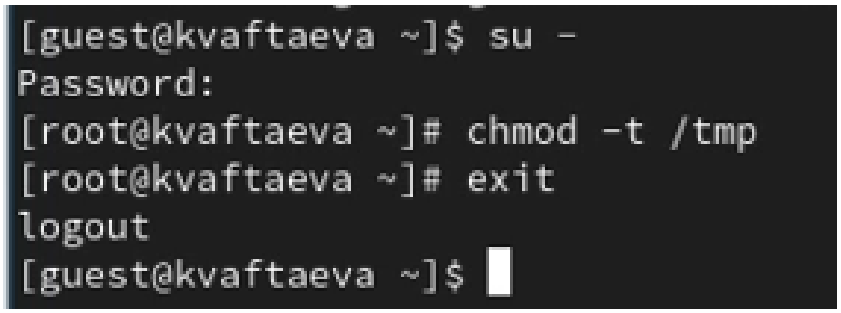

Рис. 4.11: Снятие Sticky-бита

12. От пользователя guest2 проверила, что атрибута t у директории /tmp нет командой `ls -l / | grep tmp` (рис. 4.12).

13. Повторила предыдущие шаги (рис. 4.12). Теперь мы смогли удалить файл.

14. Нам удалось удалить файл, когда мы сняли Sticky-бит.

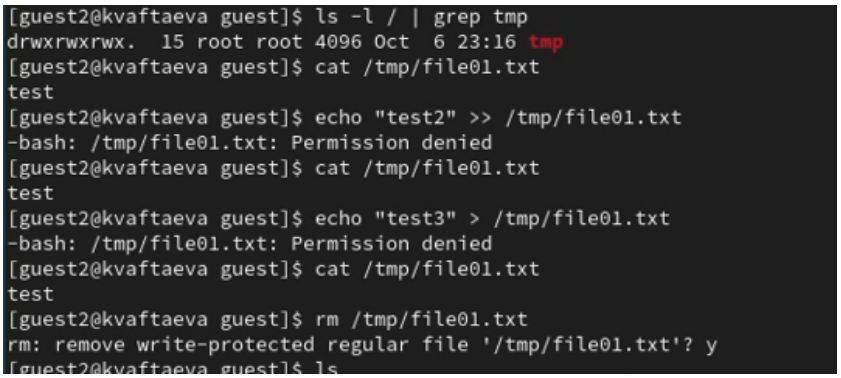

Рис. 4.12: Проверка возможности действий при отсутствии Sticky-бита

15. Повысила свои права до суперпользователя и вернула атрибут t на директорию /tmp (рис. 4.13).

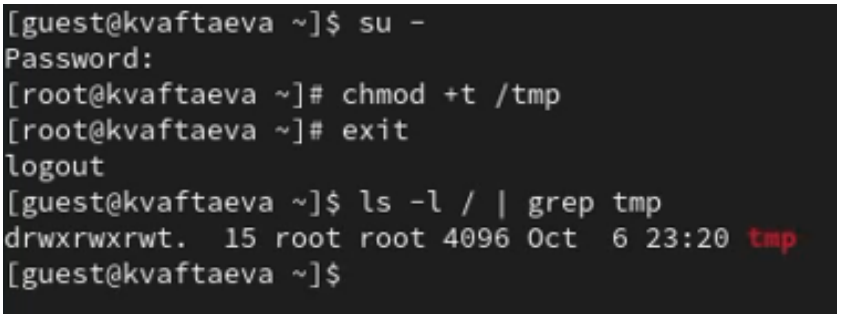

Рис. 4.13: Установка Sticky-бита

5 Выводы

Я изучила механизмы изменения, применения SetUID- и Sticky-битов. Получила практические навыки работы в консоли с дополнительными атрибутами. Рассмотрела работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

Список литературы

1. Права доступа к файлам в Linux [Электронный ресурс]. 2020. URL: https://losst.pro/prava-dostupa-k-fajlam-v-linux?ysclid=lmm5ol8ntj402722645.
2. Права доступа и владельцы в Linux [Электронный ресурс]. 2023. URL: https://hmarketing.ru/blog/server/prava-dostupa-i-vladeltsy-v-linux/?ysclid=lmm60033d6993040958.
3. Разрешения [Электронный ресурс]. 2020. URL: https://net-cheatsheets.gitbook.io/linux/razresheniya.
4. Терминал Linux. Права доступа к каталогам и файлам в Linux, команды chmod и chown [Электронный ресурс]. 2017. URL: https://linuxrussia.com/terminal-chmod-chown.html?ysclid=lng64zgf1444999917.