Отчёт по лабораторной работе №5
Информационная безопасность

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Выполнила: Абдуллина Ляйсан Раисовна,
НПИбд-01-21, 1032216538

Содержание

Список иллюстраций

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

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. [1]

- **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 относиться 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, rwt — это rw- или rwx? 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. [2]

Выполнение лабораторной работы

5.2.1. Подготовка лабораторного стенда

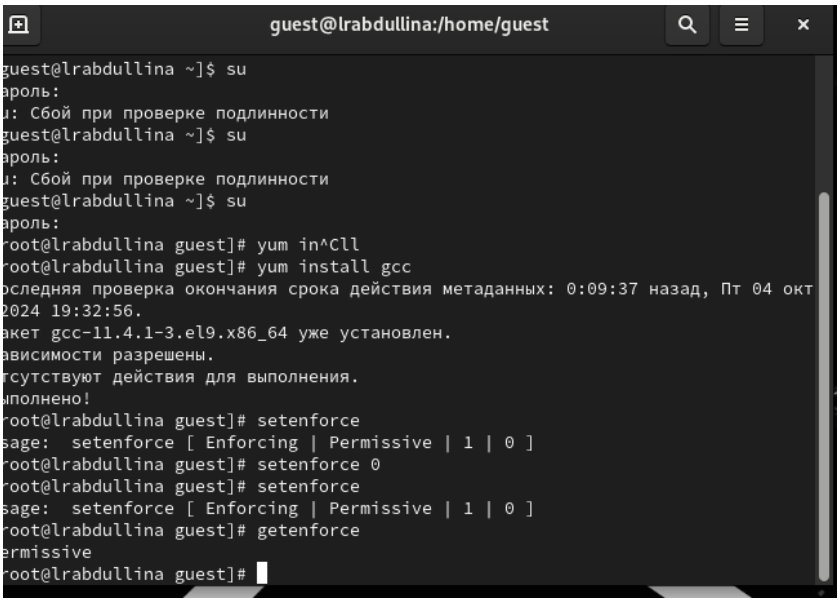

Рис. 1: (рис. 1. Установка gss)

5.3.1 Создание программы

1. Войдите в систему от имени пользователя guest.

2. Создайте программу simpleid.c.

```
[guest@lrabdullina dir1]$ touch impleid.c
[guest@lrabdullina dir1]$ nano simpleid.c
[guest@lrabdullina dir1]$ ls simpleid.c
simpleid.c
[guest@lrabdullina dir1]$ 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@lrabdullina dir1]$
```

Рис. 2: (рис. 2. simpleid.c)

3. Скомплилируйте программу и убедитесь, что файл программы создан: gcc simpleid.c -o simpleid

4. Выполните программу simpleid: ./simpleid

5. Выполните системную программу id: id и сравните полученный вами результат с данными предыдущего пункта задания.

```
}
[guest@lrabdullina dir1]$ gcc simpleid.c -o simpleid
[guest@lrabdullina dir1]$ ./simpleid
uid=1001, gid=1001
[guest@lrabdullina dir1]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined
_r:unconfined_t:s0-s0:c0.c1023
[guest@lrabdullina dir1]$
```

Рис. 3: (рис. 3. 3-5 пункты задания лабораторной)

6. Усложните программу, добавив вывод действительных идентификаторов.

```
[guest@lrabdullina dir1]$ touch simpleid2.c
[guest@lrabdullina dir1]$ nano simpleid2.c
[guest@lrabdullina dir1]$ ls simpleid2.c
simpleid2.c
[guest@lrabdullina dir1]$ 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@lrabdullina dir1]$
```

Рис. 4: (рис. 4. simpleid2.c)

7. Скомпилируйте и запустите simpleid2.c: gcc simpleid2.c -o simpleid2 ./simpleid2

```
[guest@lrabdullina dir1]$ gcc simpleid2.c -o simpleid2
[guest@lrabdullina dir1]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
```

Рис. 5: (рис. 5. 7 пункт задания лабораторной)

8. От имени суперпользователя выполните команды: chown root:guest /home/guest/simpleid2 chmod u+s /home/guest/simpleid2

9. Используйте sudo или повысьте временно свои права с помощью su. Поясните, что делают эти команды.

От имени суперпользователя выполнила команды “sudo chown root:guest /home/guest/simpleid2” и “sudo chmod u+s /home/guest/simpleid2”, затем выполнила проверку правильности установки новых атрибутов и смены владельца файла simpleid2 командой “sudo ls -l /home/guest/simpleid2” (рис. 3.9). Этими командами была произведена смена пользователя файла на root и установлен SetUID-бит.

10. Выполните проверку правильности установки новых атрибутов и смены владельца файла simpleid2: ls -l simpleid2

11. Запустите simpleid2 и id: ./simpleid2 id Сравните результаты.

12. Проделайте тоже самое относительно SetGID-бита.

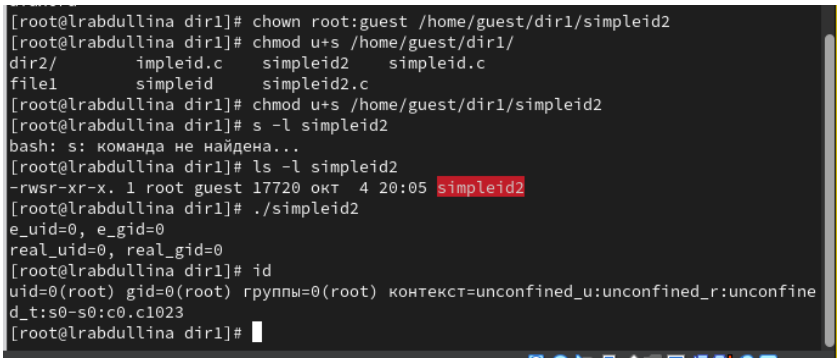

Рис. 6: (рис. 6. 8-12 пункты задания лабораторной)

13. Создайте программу readfile.c

14. Откомпилируйте её. gcc readfile.c -o readfile

```
[root@lrabdullina dir1]# touch readfile.c
[root@lrabdullina dir1]# nano readfile.c
[root@lrabdullina dir1]# ls readfile.c
readfile.c
[root@lrabdullina dir1]# 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;
}
[root@lrabdullina dir1]# gcc readfile.c -o readfile
[root@lrabdullina dir1]#
```

Рис. 7: (рис. 7. readfile.c)

15. Смените владельца у файла readfile.c (или любого другого текстового файла в системе) и измените права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог.

```
[root@lrabdullina dir1]# chown root:guest readfile
[root@lrabdullina dir1]# chown 700 readfile
[root@lrabdullina dir1]# chown -r readfile.c
chown: неверный ключ — «r»
По команде «chown --help» можно получить дополнительную информацию.
[root@lrabdullina dir1]# chmod -r readfile.c
[root@lrabdullina dir1]# chmod u+s readfile
[root@lrabdullina dir1]#
```

Рис. 8: (рис. 8. chmod)

16. Проверьте, что пользователь guest не может прочитать файл readfile.c.

17. Смените у программы readfile владельца и установите SetU’D-бит.

18. Проверьте, может ли программа readfile прочитать файл readfile.c?

19. Проверьте, может ли программа readfile прочитать файл /etc/shadow? Отразите полученный результат и ваши объяснения в отчёте.

```
[guest@lrabdullina dir1]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@lrabdullina dir1]$
```

Рис. 9: (рис. 9. 16-19 пункты Guest)

От имени суперпользователя все команды удается выполнить.

```
[guest@lrabdullina dir1]$ su
Пароль:
[root@lrabdullina dir1]# 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;
```

Рис. 10: (рис. 10. 16-18 пункты суперпользователь)

Рис. 11: (рис. 11. 19 пункт суперпользователь)

5.3.2. Исследование Sticky-бита

1. Выясните, установлен ли атрибут Sticky на директории /tmp, для чего выполните команду ls -l / | grep tmp

2. От имени пользователя guest создайте файл file01.txt в директории /tmp со словом test: echo “test” > /tmp/file01.txt

3. Просмотрите атрибуты у только что созданного файла и разрешите чтение и запись для категории пользователей «все остальные»: ls -l /tmp/file01.txt chmod o+rw /tmp/file01.txt ls -l /tmp/file01.txt

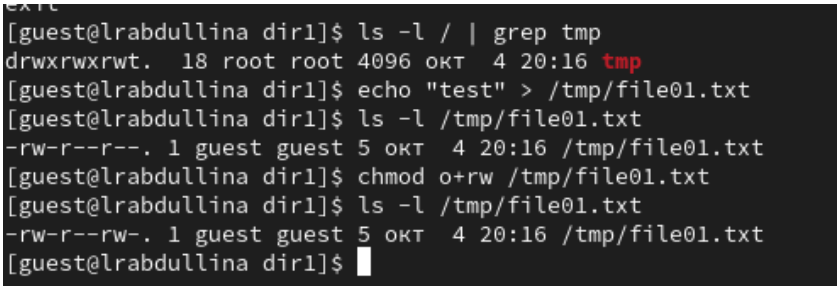

Рис. 12: (рис. 12. 1-3 пункты)

4. От пользователя guest2 (не являющегося владельцем) попробуйте прочитать файл /tmp/file01.txt: cat /tmp/file01.txt

5. От пользователя guest2 попробуйте дозаписать в файл /tmp/file01.txt слово test2 командой echo “test2” > /tmp/file01.txt

Удалось ли вам выполнить операцию? Нет.

6. Проверьте содержимое файла командой cat /tmp/file01.txt

7. От пользователя guest2 попробуйте записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию командой echo “test3” > /tmp/file01.txt

Удалось ли вам выполнить операцию? Нет.

8. Проверьте содержимое файла командой cat /tmp/file01.txt

9. От пользователя guest2 попробуйте удалить файл /tmp/file01.txt командой rm /tmp/fileOl.txt

Удалось ли вам удалить файл? Нет.

10. Повысьте свои права до суперпользователя следующей командой su и выполните после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp: chmod -t /tmp

11. Покиньте режим суперпользователя командой exit

12. От пользователя guest2 проверьте, что атрибута t у директории /tmp нет: ls -l / | grep tmp

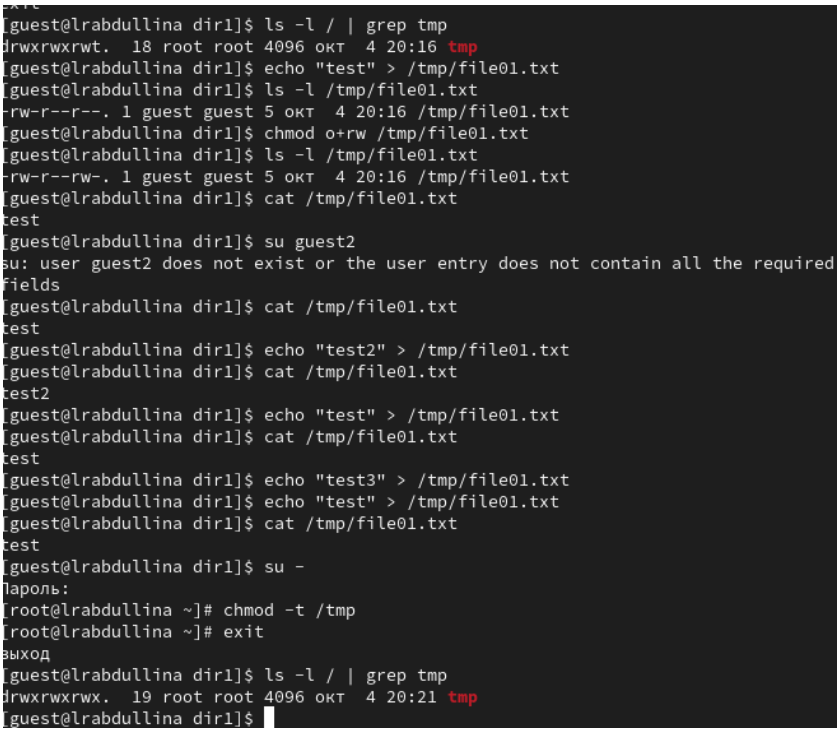

Рис. 13: (рис. 13. 4-12 пункты)

13. Повторите предыдущие шаги. Какие наблюдаются изменения?

При повторении всё получилось.

14. Удалось ли вам удалить файл от имени пользователя, не являющегося его владельцем? Удалось.

15. Повысьте свои права до суперпользователя и верните атрибут t на директорию /tmp: su chmod +t /tmp exit

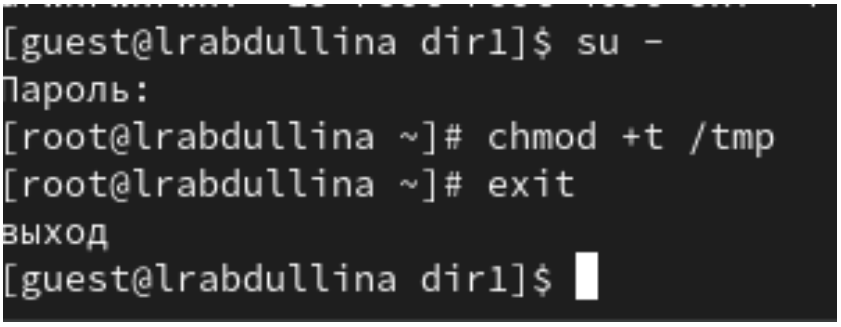

Рис. 14: (рис. 14. Возвращение атрибута)

Вывод

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

Список литературы. Библиография

[0] Методические материалы курса

[1] Дополнительные атрибуты: https://tokmakov.msk.ru/blog/item/141

[2] Компилятор GSS: http://parallel.imm.uran.ru/freesoft/make/instrum.html