Лабораторная работа 5

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Лушин Артём Андреевич

Содержание

1 Цель работы **4**

2 Выполнение лабораторной работы. Созданте программы **5**

3 Исследование Sticky-бита. **13**

4 ВАЖНОЕ ПРИМЕЧАНИЕ **16**

5 Выводы **17**

Список иллюстраций

2.1 Первая программа 5
2.2 Компиляция первой программы 5
2.3 Запуск первой программы 6
2.4 Вторая программа 6
2.5 Запуск второй программы 6
2.6 Изменение прав для root 7
2.7 Проверка работы для root 7
2.8 Установка SetUID-бита 8
2.9 Программа readfile 8
2.10 Компиляция readfile 9
2.11 Проверка на root u guest пользователях 10
2.12 Смена владельца 11
2.13 Запуск с guest 11
2.14 Запуск с root 12

3.1 Проверка наличия атрибута 13
3.2 Выдача прав для файла 13
3.3 Проверка от второго пользователя 14
3.4 Проверка без атрибута 14
3.5 Возвращение атрибута 15

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 Выполнение лабораторной работы. Созданте программы

1) Я создал файл “simpleid.c” и внёс в него программу.

```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;
}
```

Рис. 2.1: Первая программа

2) Скомпилировал программу и убедился, что файл создан правильно.

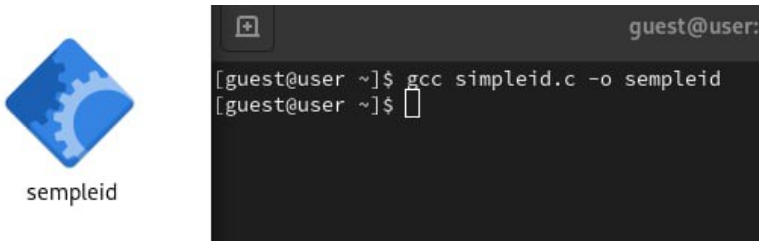

Рис. 2.2: Компиляция первой программы

3) Запустил программу и посмотрел, как она работает. Затем прописал команду “id”, чтобы сравнить данные. Все данные сходятся.

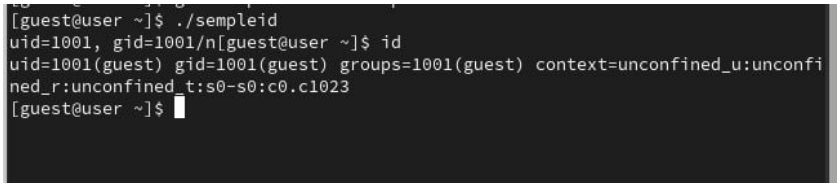

Рис. 2.3: Запуск первой программы

4) Создал второй файл и назвал его “simlpeid2.c”. Усложнил первую программу и внёс ее в файл.

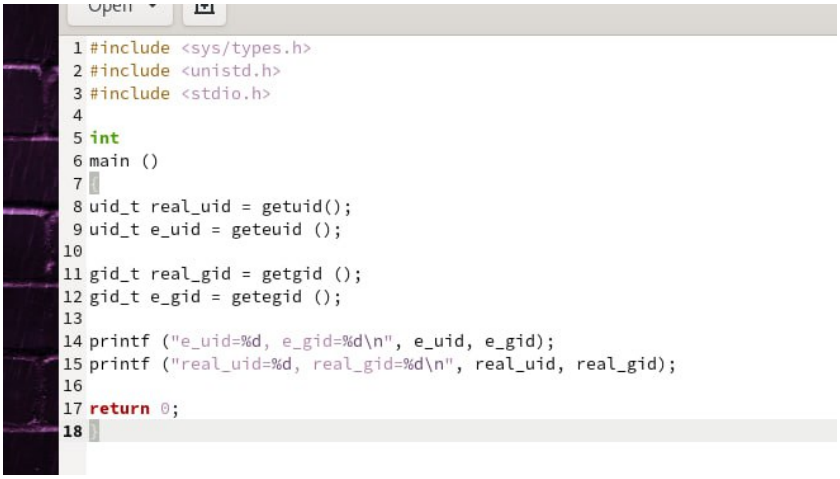

Рис. 2.4: Вторая программа

5) Скомпилировал и посмотрел вторую программу. Проверил как она работает.

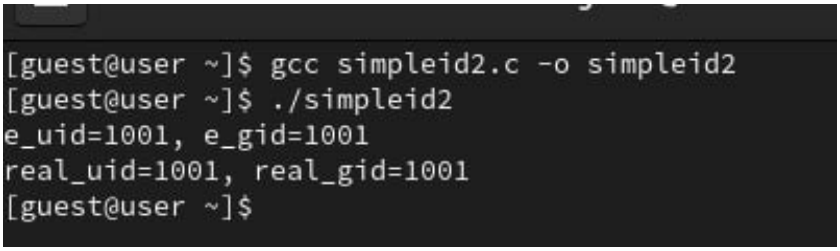

Рис. 2.5: Запуск второй программы

6) От имени суперпользователя я выполнил команды и временно повысил свои права. Команды сменили пользователя файла на root и установили SetUID-бит. Я запустил файл от имени root-пользователя и проверил сходство с командой “id”.

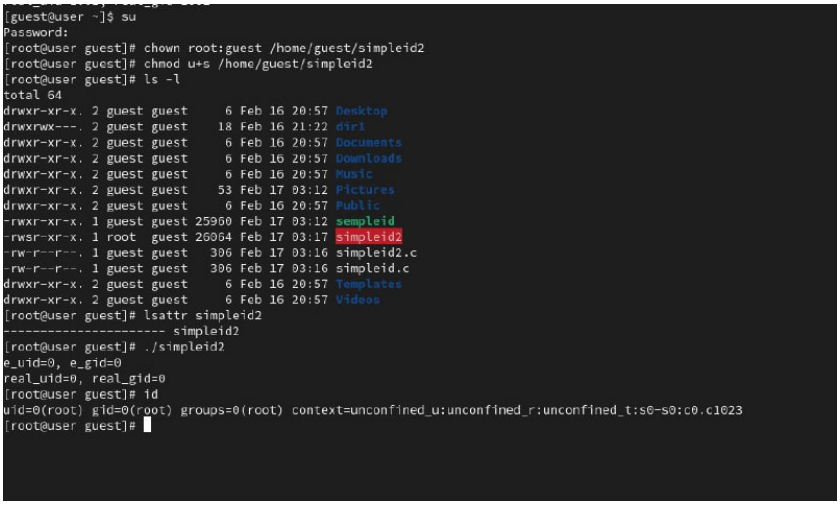

Рис. 2.6: Изменение прав для root

Рис. 2.7: Проверка работы для root

Рис. 2.8: Установка SetUID-бита

7) Я создал файл “readfile.c”. Внёс туда программу.

```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. 2.9: Программа readfile

8) Скомпилировал программу readfile.

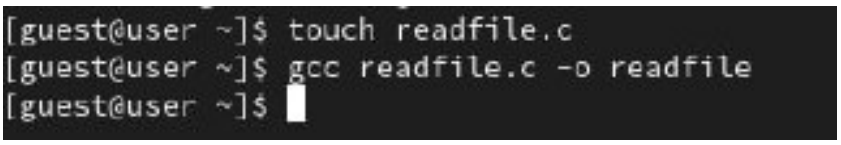

Рис. 2.10: Компиляция readfile

9) Я выдал программе “readfile” права так, чтобы root пользователь мог прочитать файл, а простой пользователь нет.

Рис. 2.11: Проверка на root u guest пользователях

10) Я сменил владельца программы “readfile” на root-пользователя.

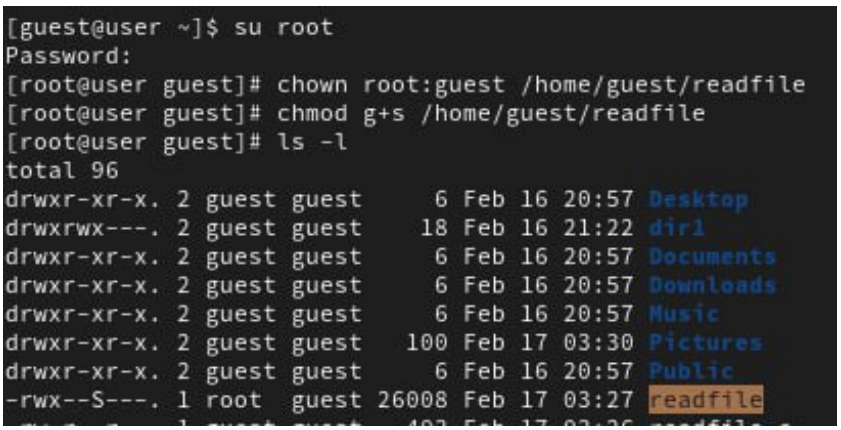

Рис. 2.12: Смена владельца

11) Попытался запустить программу и прочитать два файла с простого пользователя, но программа выдала ошибку. А если запускать с аккаунта root, то программа запускается нормально и работает. Связано это с тем, что владельцем программы является root-пользователь, а у других пользователей нет доступа и прав на использование программы.

Рис. 2.13: Запуск с guest

Рис. 2.14: Запуск с root

3 Исследование Sticky-бита.

1) Я выяснил, установлен ли атрибут Sticky (t) на директории “/tmp”. Атрибут установлен.

Рис. 3.1: Проверка наличия атрибута

2) От пользователя “guest” я создал файл “file01.txt” в директории “/tmp”. Вписал в файл слово “test”. И дал права на чтение и запись для категории “все остальные (o)”.

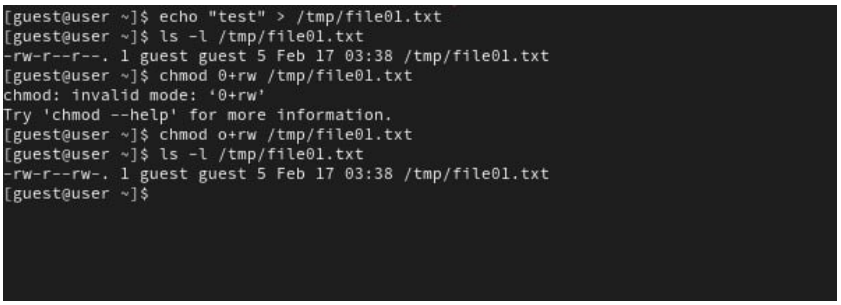

Рис. 3.2: Выдача прав для файла

3) От пользователя “guest2”, который не является владельцем, я попробовал прочитать файл. Я могу прочитать файл. Но не могу дописывать содержимое, вписывать новое или удалять этот файл.

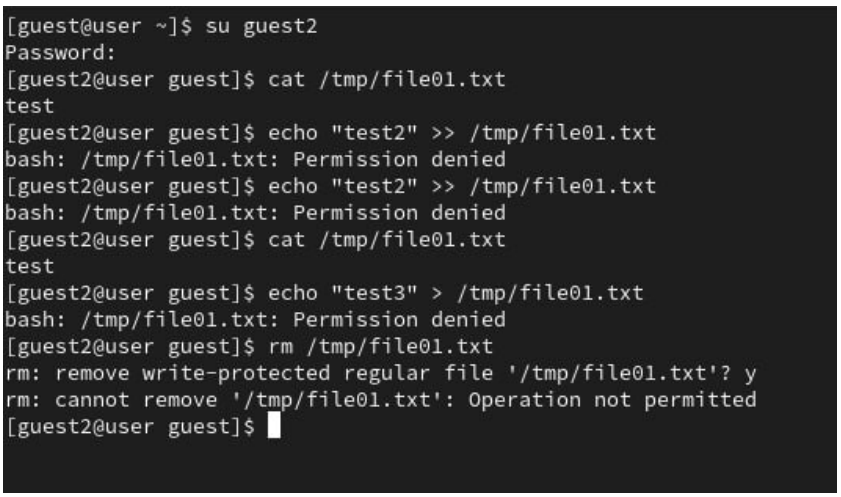

Рис. 3.3: Проверка от второго пользователя

4) я отключил атрибут “t” у директории “/tmp”. Попробовал повторить все предыдущие действия. Я так же не смог вписать в файл данные или дописать их. Но смог прочитать файл и удалить его.

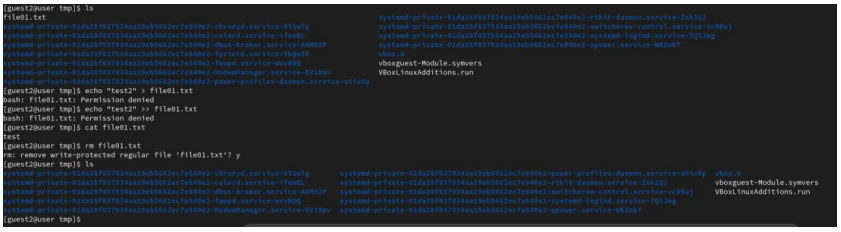

Рис. 3.4: Проверка без атрибута

5) Чтобы в дальнейшем у меня не было проблем в работе с директорией “/tmp” я вернул атрибут на директорию, используя суперпользователя.

Рис. 3.5: Возвращение атрибута

4 ВАЖНОЕ ПРИМЕЧАНИЕ

По итогам лабораторной работы я понял, что Sticky-бит создан для защиты файла от удаления. Даже не смотря на то, что я дал права на запись и чтение файлов для категории “все остальные”, я не смог вписать в файл данные с пользователя “guest2”. А не смог я это сделать, так как этот аккаунт у меня находится в группе с “guest”. То есть я не дал права на вписывание для категории “группа”, но дал права для категории “все остальные”. Из-за этого Sticky-бит не влиял на возможность записи, а влиял только на возможность удаления. А изменять файл я не мог, так как мой аккаунт находился не в той группе. Если бы я использовал другой аккаунт, который не находится в группе, результаты бы были другие.

5 Выводы

Я изучил механизмы изменения идентификатора, применил SetUID-бит и Sticky-бит. Получил практические навыки работы в консоли с дополнительными атрибутами. Рассмотрел работы механизма смены идентификатора процессов пользователя, а так же влияние бита Sticky на запись и удаление файлов.