Отчет по лабораторной работе №5

Основы информационной безопасности

Мажитов Магомед Асхабович, НКНбд-01-21

Содержание

# 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 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. [**u?**]

**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

1. Компилятор GCC

GСС - 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 c расширением .o считаются объектными [**gcc?**].

# 3 Выполнение лабораторной работы

**1** Проверил установлен ли компилятор *gcc* и *g++*.

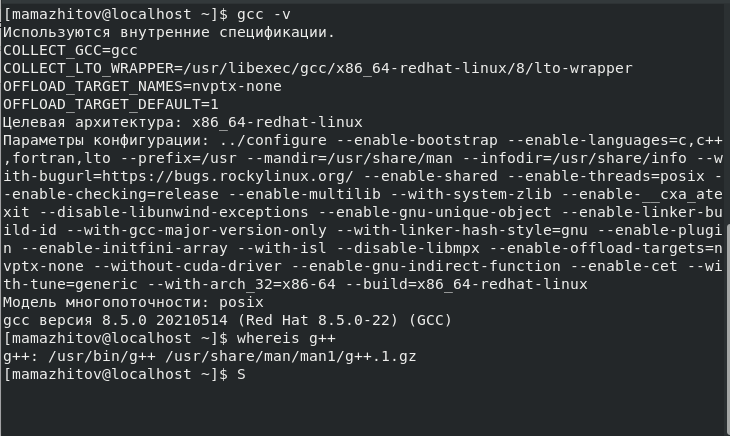


Figure 1: Компилятор

**2** Вошел в систему от имени пользователя guest и создал программу simpleid.c.

Figure 2: Создание simpleid.c

Figure 2: Создание 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;  
}

**3** Скомплилировал программу и убедился, что файл программы создан. Далее запустил исполнительный файл, а также ввел системную программу *id* для дальнейшего сравнения выводов.

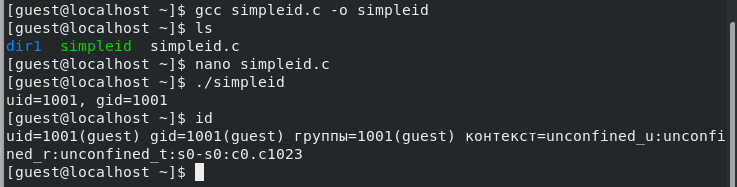


Figure 3: Компиляция simpleid.c

Результаты идентичны.

**4** Создал программу *simpleid2.c*.

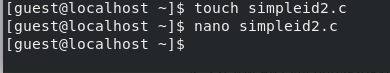


Figure 4: Создание simpleid2.c

#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;  
}

**5** Скомпилоровал программу и сравнил выводы прошлой и новой программ.

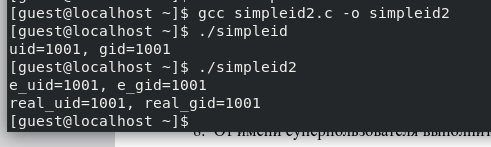


Figure 5: Сравнение

**6** Далее я поменял владельца файла *simpleid2* и изменил права доступа к нему.

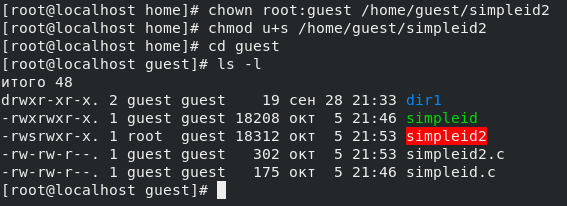


Figure 6: Манипуляции simpleid2

**7** Запустил *simpleid2* и *id*.

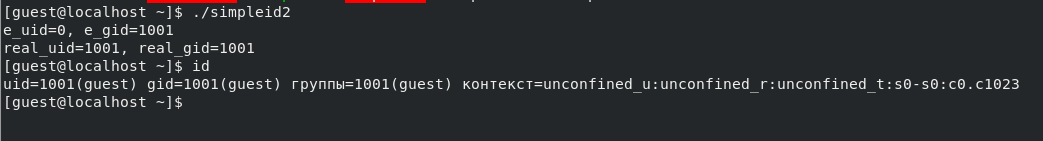


Figure 7: Сравнение

Как мы видим после изменения владельца *simpleid2*, вывод программы изменился.

**8** Создал 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;  
}

Скомпилировал файл и далее также изменил владельца *readfile* и права доступа к нему, так, чтобы только суперпользователь(root) мог прочитать его, a guest не мог.

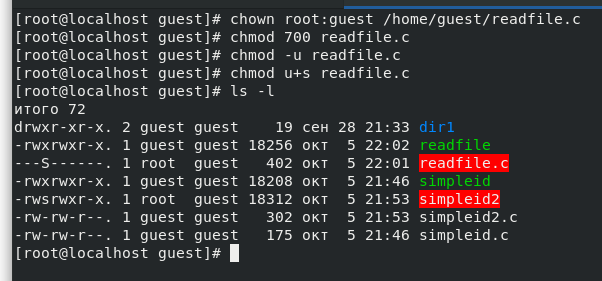


Figure 8: Изменение владельца readfile

**9** Попробовал прочитать файл от имени *guest*.

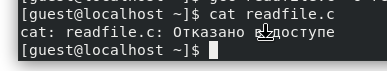


Figure 9: Попытка прочитать readfile

Попытка не увенчалась успехом.

**10** Проверил, может ли программа readfile прочитать файл *readfile.c* и */etc/shadow*.

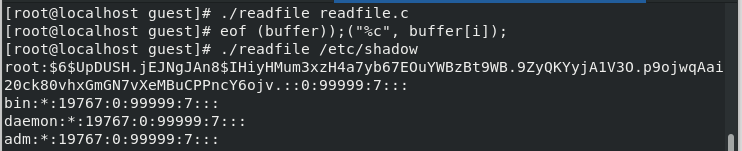


Figure 10: Попытка запустить readfile

**11** Проверил, установлен ли атрибут *Sticky* на директории /tmp.

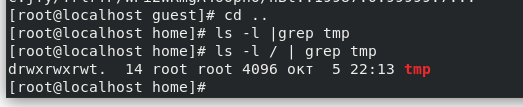


Figure 11: Проверка наличия Sticky атрибута

**12** От имени пользователя *guest* создал файл *file01.txt* в директории */tmp* со словом test и изменил права доступа к нему.

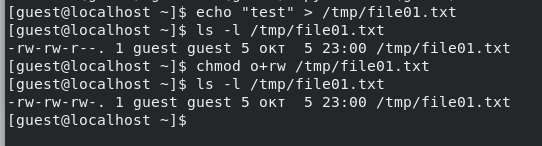


Figure 12: Создание file01.txt

**13** От пользователя guest2 (не являющегося владельцем) попробовал прочитать файл, переписать содержимое файла, а также дописать в файл новые данные.

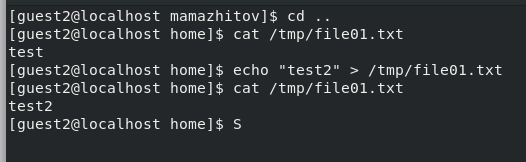


Figure 13: Манипуляции с file01.txt

**14** Попробовал удалить файл.

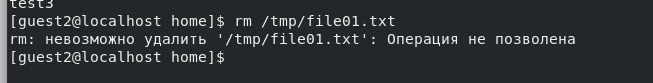


Figure 14: Попытка удалить file01.txt

**14** Повысив права до суперпользователя, снял атрибут *t*.

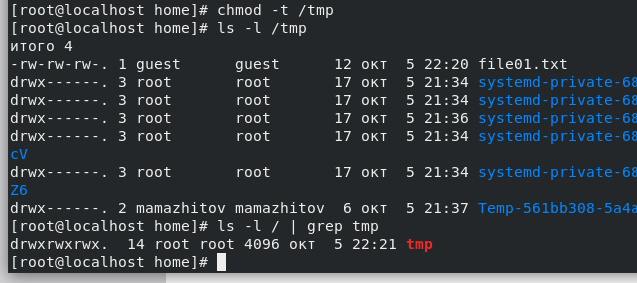


Figure 15: Снятие атрибута t

**14** Повторил действия из пунктов **13-14**.

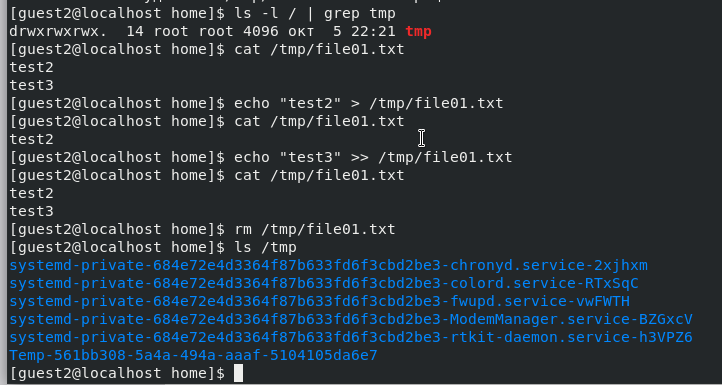


Figure 16: Манипуляции с file01.txt

В этот раз получилось удалить *file01.txt*.

**14** Попробовал удалить файл.

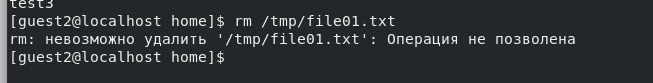


Figure 17: Попытка удалить file01.txt

# 4 Выводы

Изучил механизм изменения идентификаторов, применил SetUID- и Sticky-биты. Получил практические навыки работы в консоли с дополнительными атрибутами. Рассмотрел работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

# 5 Список литературы. Библиография