Лабораторная работа №5

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Смородова Дарья Владимировна

2022 Oct 8th

Содержание

List of Tables

List of Figures

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

Компиляторы, доступные в Linux-системах, являются частью коллекции GNU-компиляторов, известной как GCC (GNU Compiller Collection, подробнее см. http://gcc.gnu.org). В неё входят компиляторы языков C, C++, Java, Objective-C, Fortran и Chill. Будем использовать лишь первые два.

Компилятор языка C называется gcc. Компилятор языка C++ называется g++ и запускается с параметрами почти так же, как gcc. Проверить это можно следующими командами: whereis gcc и whereis g++

Первый шаг заключается в превращении исходных файлов в объектный код: gcc -c file.c

В случае успешного выполнения команды (отсутствие ошибок в коде) полученный объектный файл будет называться file.o.

Объектные файлы невозможно запускать и использовать, поэтому после компиляции для получения готовой программы объектные файлы необходимо скомпоновать. Компоновать можно один или несколько файлов. В случае использования хотя бы одного из файлов, написанных на C++, компоновка производится с помощью компилятора g++. Строго говоря, это тоже не вполне верно. Компоновка объектного кода, сгенерированного чем бы то ни было (хоть вручную), производится линкером ld, g++ его просто вызывает изнутри. Если же все файлы написаны на языке C, нужно использовать компилятор gcc.

Например, так: gcc -o program file.o

В случае успешного выполнения команды будет создана программа program (исполняемый файл формата ELF с установленным атрибутом +x).

Компилирование — это процесс. Компилятор gcc (g++) имеет множество параметров, влияющих на процесс компиляции. Он поддерживает различные режимы оптимизации, выбор платформы назначения и пр.

Также возможно использование make-файлов (Makefile) с помощью утилиты make для упрощения процесса компиляции.

Такое решение подойдёт лишь для простых случаев. Если говорить про пример выше, то компилирование одного файла из двух шагов можно сократить вообще до одного, например: gcc file.c

В этом случае готовая программа будет иметь называние a.out.

Механизм компилирования программ в данной работе не мог быть не рассмотрен потому, что использование программ, написанных на bash, для изучения SetUID- и SetGID- битов, не представляется возможным. Связано это с тем, что любая bash-программа интерпретируется в процессе своего выполнения, т.е. существует сторонняя программа-интерпретатор, которая выполняет считывание файла сценария и выполняет его последовательно. Сам интерпретатор выполняется с правами пользователя, его запустившего, а значит, и выполняемая программа использует эти права.

При этом интерпретатору абсолютно всё равно, установлены SetUID-, SetGID-биты у текстового файла сценария, атрибут разрешения запуска «x» или нет. Важно, чтобы был установлен лишь атрибут, разрешающий чтение «r».

Также не важно, был ли вызван интерпретатор из командной строки (запуск файла, как bash file1.sh), либо внутри файла была указана строчка #!/bin/bash.

Логично спросить: если установление SetUID- и SetGID- битов на сценарий не приводит к нужному результату как с исполняемыми файлами, то что мешает установить эти биты на сам интерпретатор? Ничего не мешает, только их установление приведёт к тому, что, так как владельцем /bin/bash является root: ls -l /bin/bash все сценарии, выполняемые с использованием /bin/bash, будут иметь возможности суперпользователя — совсем не тот результат, который хотелось

бы видеть. [1]

[1]Методические материалы к лабораторной работе

3 Выполнение лабораторной работы [1]

1. При помощи команды gcc -v проверим, установлен ли компилятор gcc (рис. 3.1):

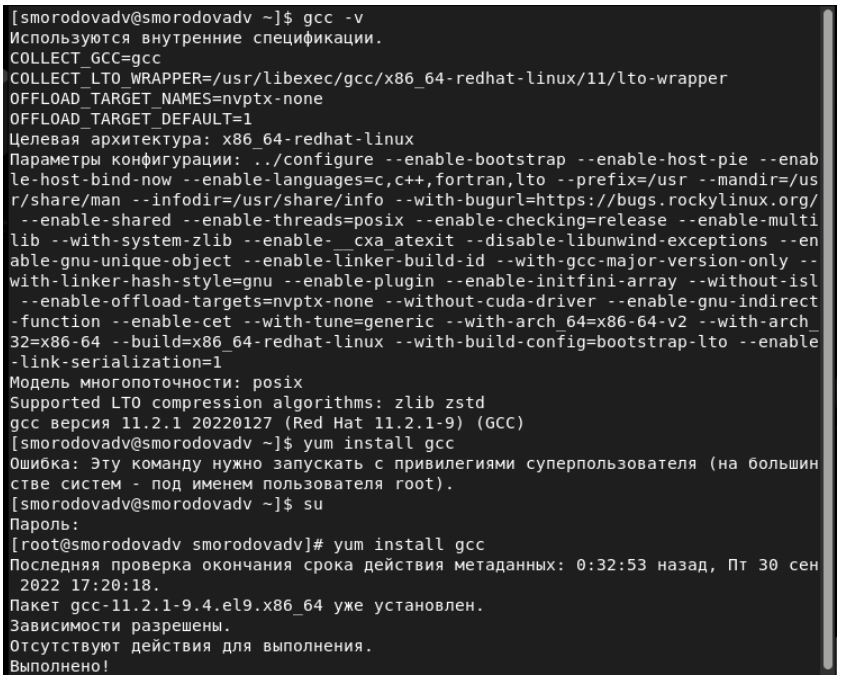

Figure 3.1: Проверка установки компилятора gcc

2. При помощи команд setenforce 0 и getenforce отключим систему запретов до очередной перезагрузки системы (рис. 3.2):

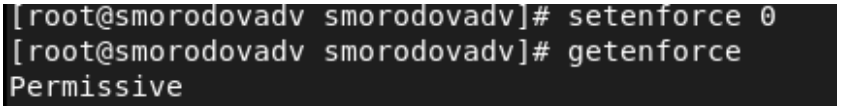

Figure 3.2: Отключение системы запретов

[1] Методические материалы к лабораторной работе

3. Проверим компиляторы языков C++ и C командами whereis gcc и whereis g++ (рис. 3.3):

Figure 3.3: Проверка компиляторов

4. Войдем в систему от имени пользователя guest и создадим программу simpleid.c (рис. 3.4 - 3.5):

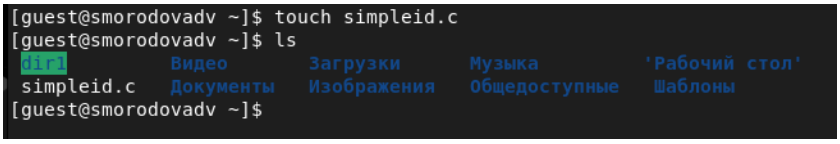

Figure 3.4: Создание файла simpleid.c

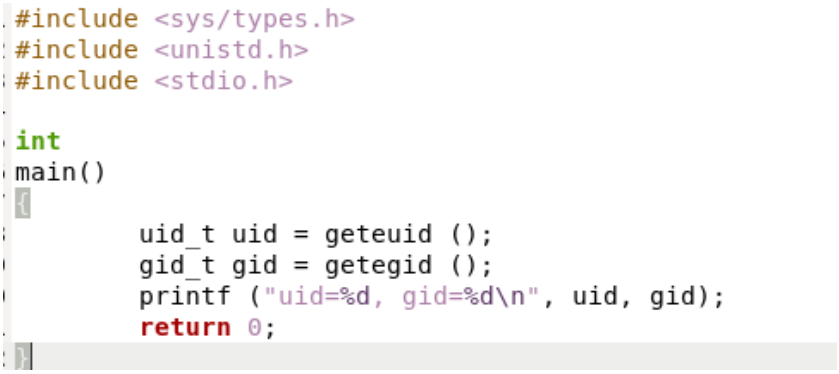

Figure 3.5: Содержимое файла simpleid.c

5. Скомплилируем программу и убедимся, что файл программы создан, а затем выполним программу simpleid (рис. 3.6):

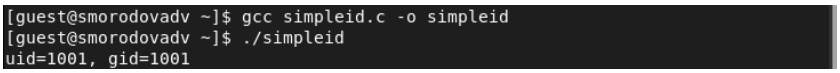

Figure 3.6: Компиляция и запуск файла simpleid.c

6. Выполните системную программу id. Результат вывода этой команды совпадает с выводом программы simpleid (рис. 3.7):

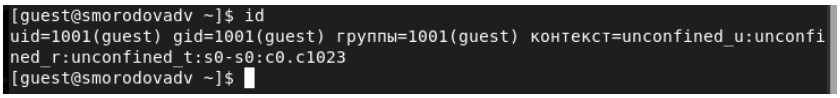

Figure 3.7: Команда id

7. Усложним программу, добавив вывод действительных идентификаторов (рис. 3.8):

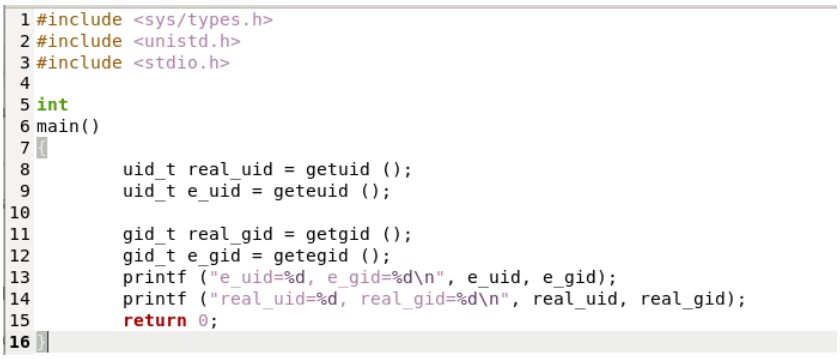

Figure 3.8: Содержимое файла simpleid2.c

8. Скомпилируем и запустим simpleid2.c (рис. 3.9):

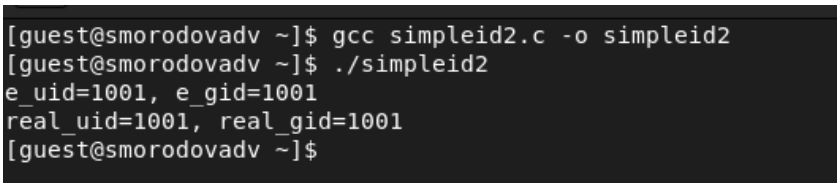

Figure 3.9: Компиляция и запуск файла simpleid2.c

9. От имени суперпользователя выполним команды chown root:guest /home/guest/simpleid2 и chmod u+s /home/guest/simpleid2 (рис. 3.10):

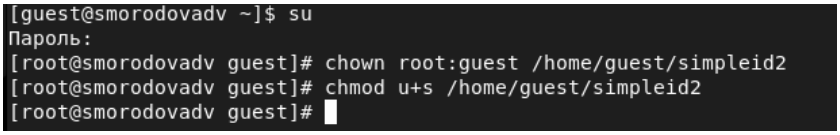

Figure 3.10: Команды chown и chmod

10. Выполним проверку правильности установки новых атрибутов и смены владельца файла simpleid2 командой ls -l simpleid2 (рис. 3.11):

```
[root@smorodovadv guest]# ls -l simpleid2
-rwsrwxr-x. 1 root guest 26008 сен 30 18:05 simpleid2
[root@smorodovadv guest]#
```

Figure 3.11: Проверка правильности установки новых атрибутов

11. Запустим simpleid2 и id. Вывод в обоих случаях совпал (рис. 3.12):

```
[root@smorodovadv guest]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@smorodovadv guest]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconf
ined_t:s0-s0:c0.c1023
[root@smorodovadv guest]#
```

Figure 3.12: Запуск simpleid2 и id

12. Проделаем тоже самое относительно SetGID-бита (рис. 3.13):

```
[root@smorodovadv guest]# chmod g+s /home/guest/simpleid2
[root@smorodovadv guest]# ls -l simpleid2
-rwsrwsr-x. 1 root guest 26008 сен 30 18:05 simpleid2
[root@smorodovadv guest]# ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=0
[root@smorodovadv guest]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconf
ined_t:s0-s0:c0.c1023
[root@smorodovadv guest]#
```

Figure 3.13: Сравнение SetGID-бита

13. Создайте программу readfile.c (рис. 3.14):

```
#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;
}
```

Figure 3.14: Содержание файла readfile.c

14. Компиляция файла readfile.c (рис. 3.15):

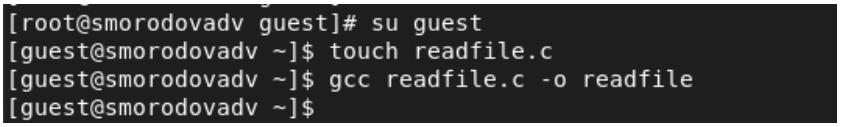

Figure 3.15: Компиляция файла readfile.c

15. Сменим владельца у файла readfile.c (или любого другого текстового файла в системе) и изменим права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог (рис. 3.16):

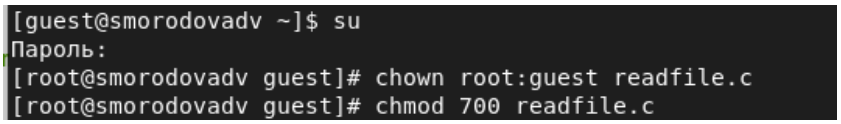

Figure 3.16: Смена владельца и прав у файла readfile.c

16. Проверим, что пользователь guest не может прочитать файл readfile.c (рис. 3.17):

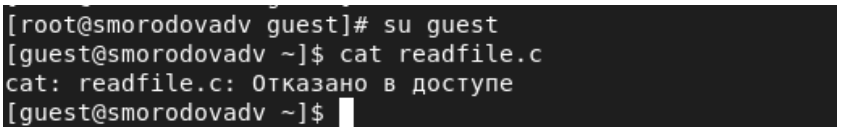

Figure 3.17: Проверка возможности прочитать файл readfile.c

17. Сменим у программы readfile владельца и установим SetU'D-бит (рис. 3.18):

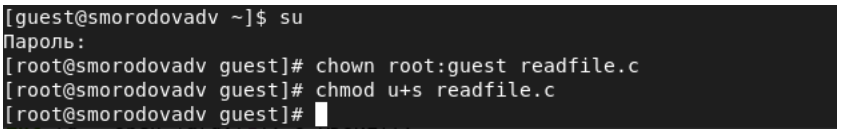

Figure 3.18: Смена у программы readfile владельца и установка SetU'D-бит

18. Проверим, может ли программа readfile прочитать файл readfile.c (рис. 3.19):

```
[root@smorodovadv guest]# ./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]);
        }
        while (bytes_read == sizeof (buffer));
        close (fd);
        return 0;
}
[root@smorodovadv guest]#
```

Figure 3.19: Проверка возможности прочитать файл readfile.c

19. Проверим, может ли программа readfile прочитать файл /etc/shadow (рис. 3.20):

```
[root@smorodovadv guest]# ./readfile /etc/shadow
root:$6$MIUJDzjA/shK/XEE$FCafDmVfJ0Wm2UlrimsoOqph1F3qk947LxjFRTe/U4fezZ19Rho3CiH
gvU4fL7ldPu8KOeD9vzIINhNJ630F2.::0:99999:7:::
bin:*:19123:0:99999:7:::
daemon:*:19123:0:99999:7:::
adm:*:19123:0:99999:7:::
lp:*:19123:0:99999:7:::
sync:*:19123:0:99999:7:::
shutdown:*:19123:0:99999:7:::
halt:*:19123:0:99999:7:::
mail:*:19123:0:99999:7:::
operator:*:19123:0:99999:7:::
games:*:19123:0:99999:7:::
ftp:*:19123:0:99999:7:::
nobody:*:19123:0:99999:7:::
systemd-coredump:!!:19245::::::
dbus:!!:19245::::::
polkitd:!!:19245::::::
rtkit:!!:19245::::::
sssd:!!:19245::::::
avahi:!!:19245::::::
pipewire:!!:19245::::::
libstoragemgmt:!!:19245::::::
tss:!!:19245::::::
geoclue:!!:19245::::::
cockpit-ws:!!:19245::::::
cockpit-wsinstance:!!:19245::::::
setroubleshoot:!!:19245::::::
flatpak:!!:19245::::::
colord:!!:19245::::::
```

Figure 3.20: Проверка возможности прочитать файл /etc/shadow

20. Выясним, установлен ли атрибут Sticky на директории /tmp командой ls -l / | grep tmp (рис. 3.21):

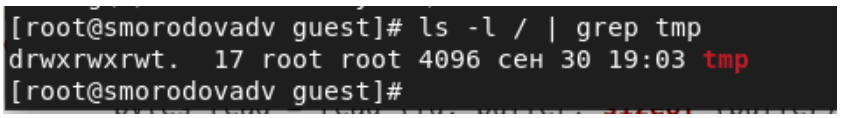

Figure 3.21: Проверка установки атрибута Sticky на директории /tmp

21. От имени пользователя guest создадим файл file01.txt в директории /tmp со словом test (рис. 3.22):

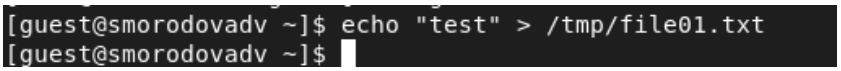

Figure 3.22: Создание файла file01.txt в директории /tmp со словом test

22. Просмотрим атрибуты у только что созданного файла и разрешим чтение и запись для категории пользователей «все остальные» (рис. 3.23):

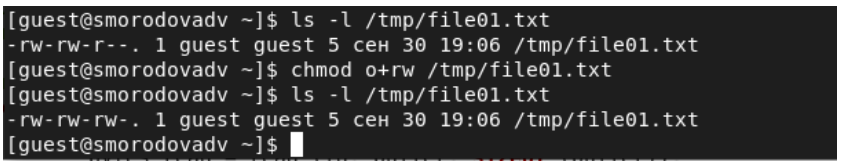

Figure 3.23: Просмотр атрибутов и разрешение чтения и записи для категории пользователей «все остальные»

23. От пользователя guest2 (не являющегося владельцем) попробуем прочитать файл /tmp/file01.txt (рис. 3.24):

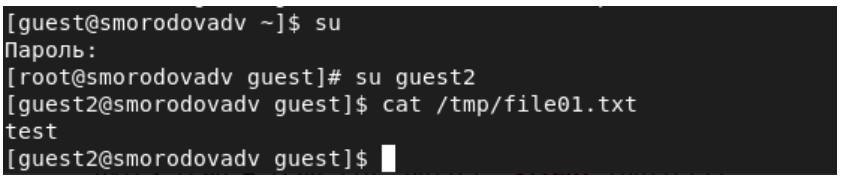

Figure 3.24: Попытка прочитать файл /tmp/file01.txt от пользователя guest2

24. От пользователя guest2 попробуем дозаписать в файл /tmp/file01.txt слово test2. Это получилось сделать. Проверим содержимое файла (рис. 3.25):

Figure 3.25: Попытка дозаписать в файл /tmp/file01.txt слово test2 от пользователя guest2

25. От пользователя guest2 попробуем записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию и проверим содержимое файла (рис. 3.26):

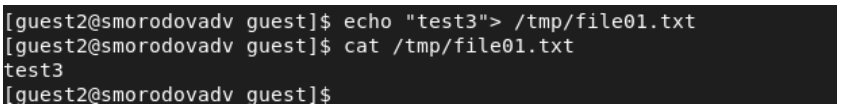

Figure 3.26: Попытка перезаписать файл /tmp/file01.txt словом test3 от пользователя guest2

26. От пользователя guest2 попробуем удалить файл /tmp/file01.txt командой rm /tmp/fileOl.txt, это сделать не удалось (рис. 3.27):

Figure 3.27: Попытка удалить файл /tmp/file01.txt от пользователя guest2

27. Повысим свои права до суперпользователя командой su - и выполним после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp командой chmod -t /tmp. Покинем режим суперпользователя командой exit (рис. 3.28):

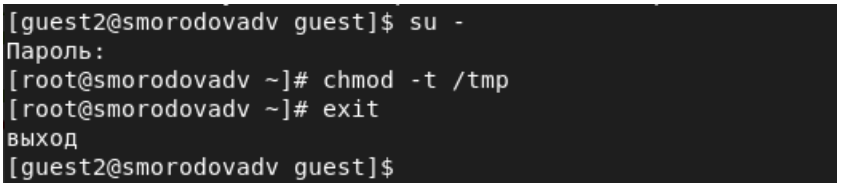

Figure 3.28: Снятие атрибута t с директории /tmp

28. От пользователя guest2 проверим, что атрибута t у директории /tmp нет (рис. 3.29):

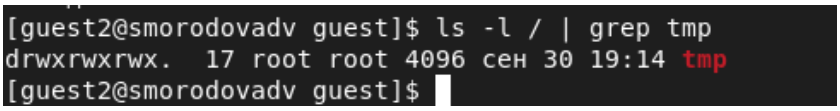

Figure 3.29: Проверка снятия атрибута t с директории /tmp

29. Повторим предудущие шаги, и увидим, что теперь мы можем не только добавлять текст файл, перезаписывать его и показывать содержимое, но и удалять файл (рис. 3.30):

Figure 3.30: Повтор предыдущих шагов

30. Повысим свои права до суперпользователя и вернем атрибут t на директорию /tmp (рис. 3.31):

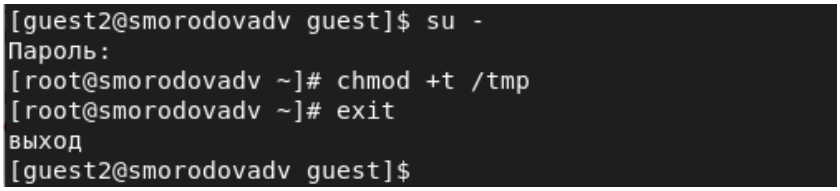

Figure 3.31: Возвращение атрибута t директории /tmp

4 Выводы

В ходе данной лабораторной работы, мы изучили механизмы изменения идентификаторов, применения SetUID- и Sticky-битов, получили практические навыки работы в консоли с дополнительными атрибутами, а также рассмотрели работу механизма смены идентификатора процессов пользователей и влияние бита Sticky на запись и удаление файлов.

5 Список литературы

1. Методические материалы к лабораторной работе, представленные на сайте “ТУИС РУДН”