Лабораторная работа №5

Информационная безопасность

Николаев Дмитрий Иванович

Содержание

Список иллюстраций

Список таблиц

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

2.1 SetUID

SetUID (сокращения от англ. set user ID upon execution — “установка ID пользователя во время выполнения”) разрешает пользователям запускать исполняемые файлы с правами владельца исполняемого файла. Иногда файлы требуют разрешения на выполнение для пользователей, которые не являются членами группы владельца, в этом случае потребуется предоставить специальные разрешения на выполнение. Когда SetUID установлен, пользователь может запускать любую программу, как её владелец.

2.2 Sticky-бит

В случае, если этот бит установлен для папки, то файлы в этой папке могут быть удалены только их владельцем. Пример использования этого бита в операционной системе это системная папка /tmp . Эта папка разрешена на запись любому пользователю, но удалять файлы в ней могут только пользователи, являющиеся владельцами этих файлов. Символ «t» указывает, что на папку установлен Sticky-бит.

3 Выполнение лабораторной работы

С помощью команды gcc -v убедимся, что у меня установлен компилятор gcc ([3.1]).

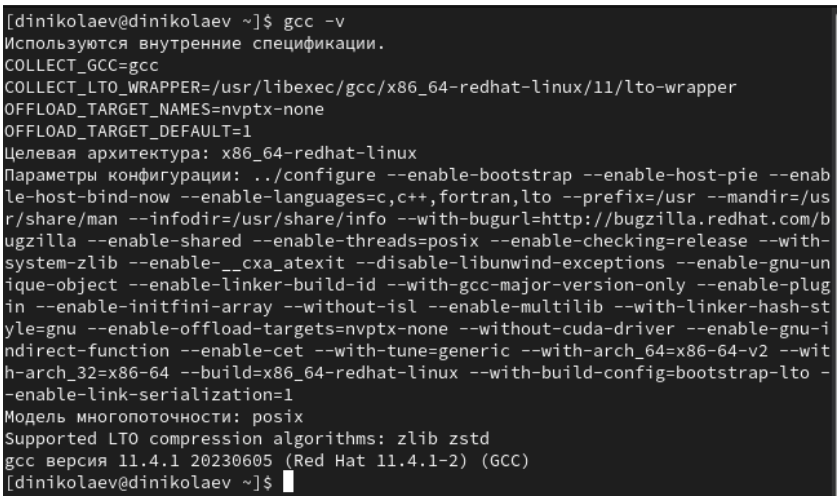


Рис. 3.1: Проверка установленности gcc

3.1 Создание программы

1. Действуя согласно [1], зайдём от имени пользователя guest и создадим файл simpleid.c ([3.2]–[3.3]).

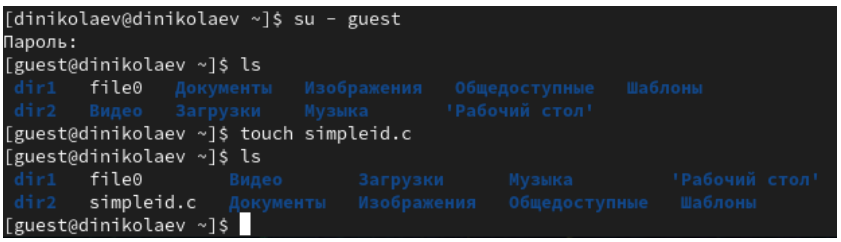


Рис. 3.2: Создание файла simpleid.c от имени guest

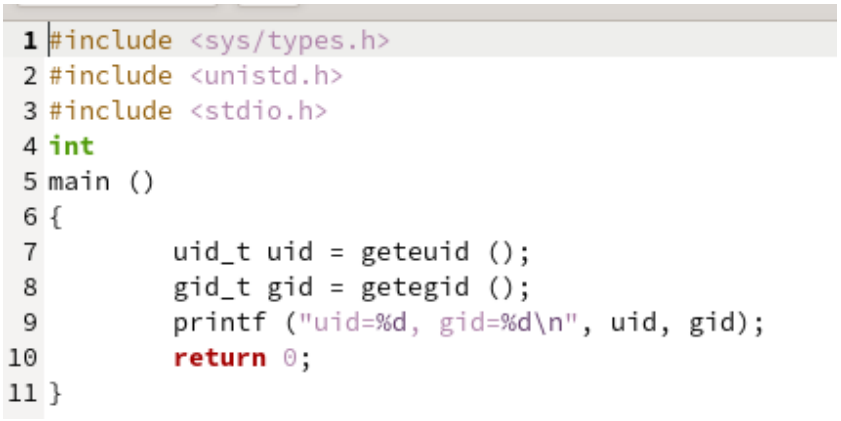


Рис. 3.3: Код файла simpleid.c

2. Скомпилируем программу и убедимся, что исполняемый файл программы создан: gcc simpleid.c -o simpleid. Выполним программу simpleid: ./simpleid и сравним с выводом системной программы id. Как можно заметить, выводы идентичны ([3.4]).

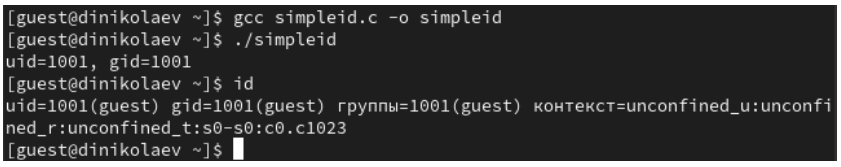


Рис. 3.4: Компиляция и запуск программы simpleid

3. Усложним программу, добавив вывод действительных идентификаторов. Получившуюся программу назовём simpleid2.c ([3.5]). После скомпилируем и запустим получившийся исполняемый файл ([3.6]).

```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;
}
```

Рис. 3.5: Код программы simpleid2.c

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

Рис. 3.6: Компиляция и исполнение файла simpleid2

4. От имени суперпользователя выполним команды ([3.7]):

   - chown root:guest /home/guest/simpleid2
   - chmod u+s /home/guest/simpleid2

```
[guest@dinikolaev ~]$ su
Пароль:
[root@dinikolaev guest]# chown root:guest /home/guest/simpleid2
[root@dinikolaev guest]# chmod u+s /home/guest/simpleid2
```

Рис. 3.7: Изменение владельца и прав доступа к файлу simpleid2

Команда chown root:guest /home/guest/simpleid2 меняет владельца файла (с guest на root). Команда chmod u+s /home/guest/simpleid2 меняет права доступа к файлу (добавляет атрибут s (вместо x у владельца файла)).

5. Выполним проверку правильности установки новых атрибутов и смены владельца файла simpleid2: ls -l simpleid2 и запустим simpleid2 и id. Для пользователя root все выводы равняются 0 ([3.8]).

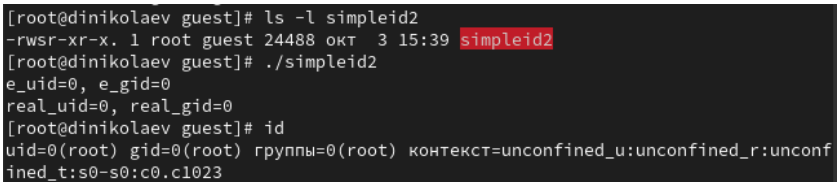


Рис. 3.8: Проверка атрибутов файла simpleid2 и вывод для root пользователя

6. Проверим вывод для пользователя guest ([3.9]).

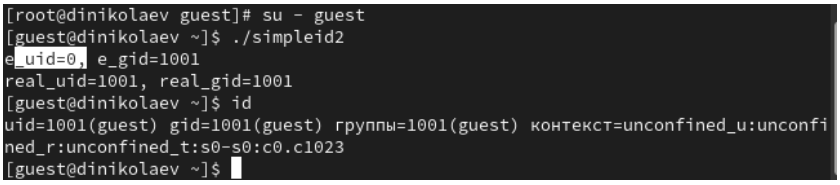


Рис. 3.9: Вывод simpleid2 для пользователя guest

7. Проделаем тоже самое относительно SetGID-бита, предварительно сняв UID-бит ([3.10]).

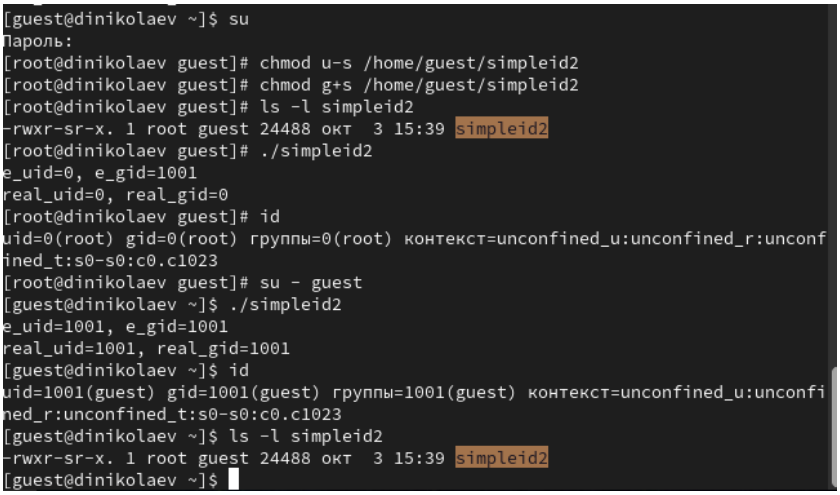
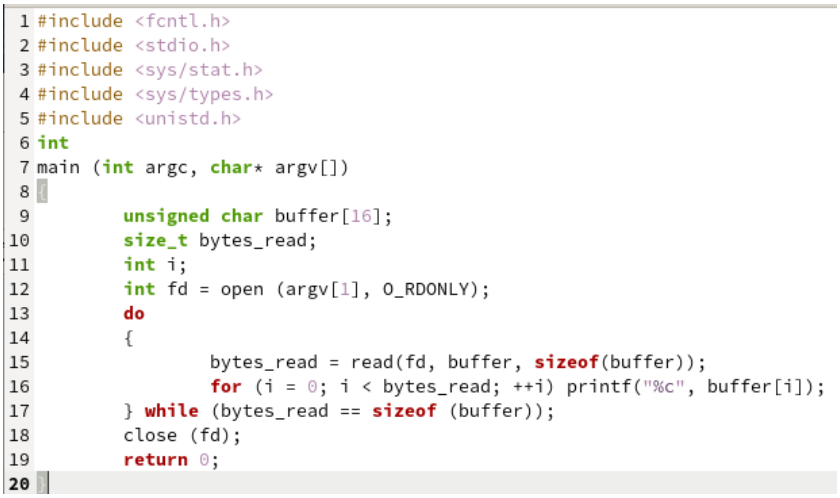


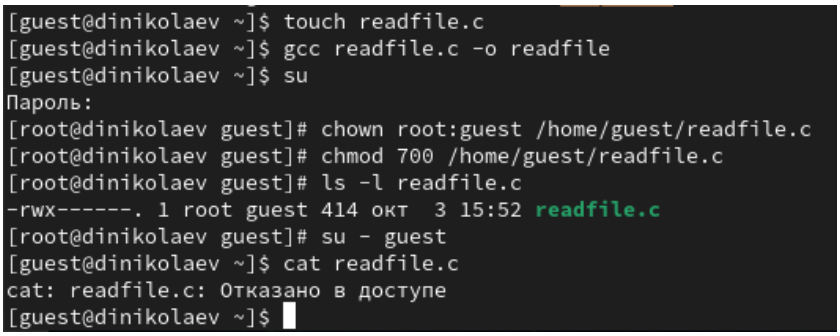
Рис. 3.10: Установка SetGID-бита и проверка вывода файла simpleid2

8. Создадим программу readfile.c ([3.11]) и скомпилируем её: gcc readfile.c -o readfile. Сменим владельца у файла readfile.c и изменим права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог, совершив проверку выполненного ([3.12]).



```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. 3.11: Код программы readfile.c



```
[guest@dinikolaev ~]$ touch readfile.c
[guest@dinikolaev ~]$ gcc readfile.c -o readfile
[guest@dinikolaev ~]$ su
Пароль:
[root@dinikolaev guest]# chown root:guest /home/guest/readfile.c
[root@dinikolaev guest]# chmod 700 /home/guest/readfile.c
[root@dinikolaev guest]# ls -l readfile.c
-rwx------. 1 root guest 414 окт  3 15:52 readfile.c
[root@dinikolaev guest]# su - guest
[guest@dinikolaev ~]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@dinikolaev ~]$
```

Рис. 3.12: Смена владельца и прав на файл readfile.c с проверкой

9. Сменим у программы readfile владельца и установим SetUID-бит ([3.13]).

```
[guest@dinikolaev ~]$ su
Пароль:
[root@dinikolaev guest]# chown root:guest /home/guest/readfile
[root@dinikolaev guest]# chmod u+s /home/guest/readfile
[root@dinikolaev guest]# ls -l readfile
-rwsr-xr-x. 1 root guest 24432 окт  3 15:52 readfile
[root@dinikolaev guest]# su - guest
```

Рис. 3.13: Смена владельца и установление SetUID-бита на файл readfile

10. Проверим, может ли программа readfile прочитать файл readfile.c ([3.14]) и /etc/shadow ([3.15]). Так как мы установили SetUID-бит программе readfile, то ей временно предоставляются права владельца файла.

```
[guest@dinikolaev ~]$ ./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;
}
```

Рис. 3.14: Чтение программы readfile.c с помощью readfile

Рис. 3.15: Чтение программы/etc/shadow с помощью readfile

3.2 Исследование Sitcky-бита

1. Выясним, установлен ли атрибут Sticky на директории /tmp, для чего выполним команду ls -l / | grep tmp. От имени пользователя guest создадим файл file01.txt в директории /tmp со словом test: echo “test” > /tmp/file01.txt. Просмотрим атрибуты у только что созданного файла и разрешим чтение и запись для категории пользователей «все остальные» ([3.16]):

    - ls -l /tmp/file01.txt

- chmod o+rw /tmp/file01.txt
- ls -l /tmp/file01.txt

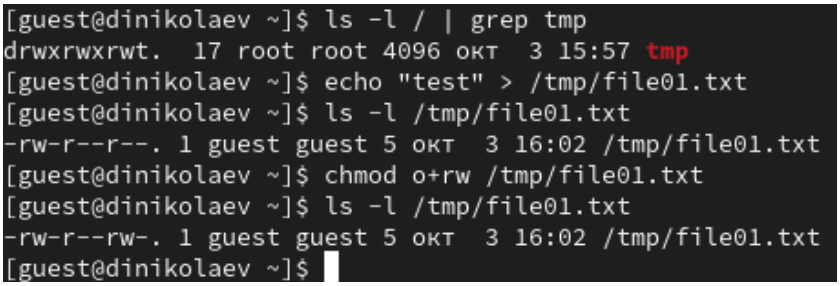


Рис. 3.16: Проверка Sticky-бита и создание файла file01.txt с правами на чтение и запись

Установим права на запись и чтение для всех категорий пользователей ([3.17]).

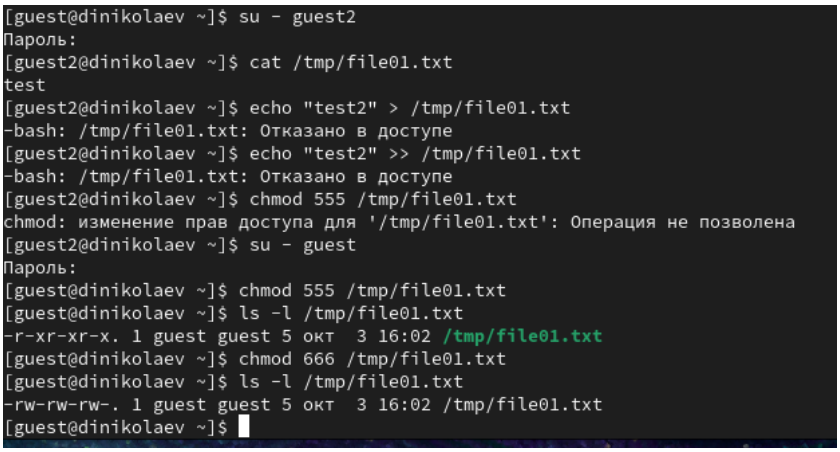


Рис. 3.17: Установка прав на запись и чтение file01.txt

2. От имени пользователя guest2 (не являющегося владельцем) попробуем прочитать файл /tmp/file01.txt: cat /tmp/file01.txt, дозаписать текст в него командой echo “test2” » /tmp/file01.txt, перезаписать файл командой echo “test3” > /tmp/file01.txt и попробуем удалить файл. Все действия кроме удаления выполнить удалось ([3.18]).

Рис. 3.18: Проверка некоторых действий с файлом file01.txt от имени пользователя guest2

3. Снимем атрибут t (Sticky-бит) с файла file01.txt от имени суперпользователя и повторим действия из предыдущего шага ([3.19]). Помимо уже успешно выполнимых действий с файлом, мы теперь смогли удалить файл от имени пользователя, не являющегося его владельцем. Таким образом, Sticky-bit позволяет защищать файлы от случайного удаления, когда несколько пользователей имеют права на запись в один и тот же файл. Если у файла атрибут t стоит, значит пользователь может удалить файл, только если он является пользователем-владельцем файла или каталога, в котором содержится файл. Если же этот атрибут не установлен, то удалить файл могут все пользователи, которым позволено удалять файлы из каталога.

Рис. 3.19: Проверка некоторых действий с файлом file01.txt без Sticky-бита от имени пользователя guest2

4. Повысим свои права до суперпользователя и вернём атрибут t на директорию /tmp ([3.20]) и проверим возвращение атрибута ([3.21]).

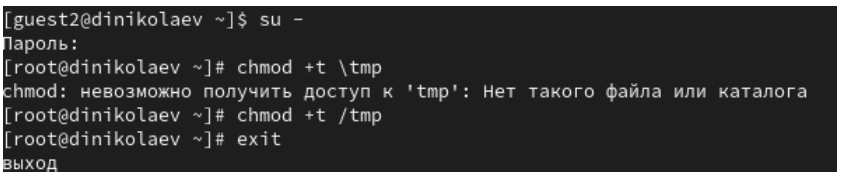


Рис. 3.20: Возвращение Sticky-бита на директории /tmp

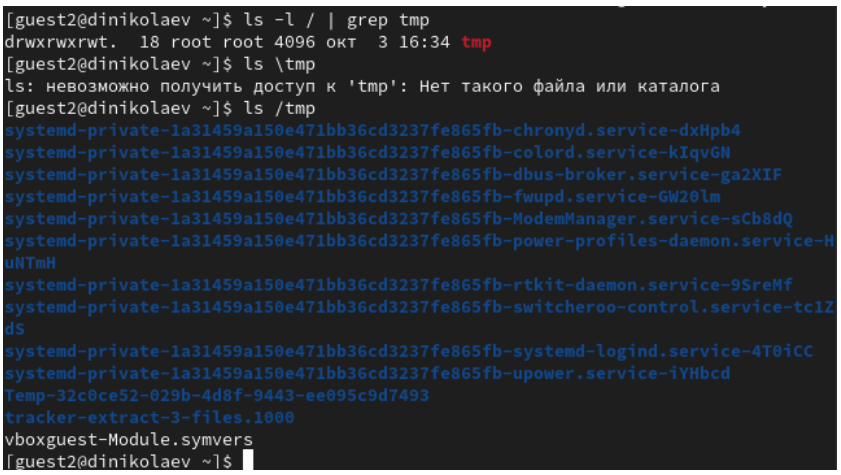


Рис. 3.21: Проверка атрибутов на директории /tmp

4 Выводы

В ходе выполнения лабораторной работы я изучил механизмы изменения идентификаторов, применения SetUID и Sticky-битов. Я получила практические навыки работы в консоли с дополнительными атрибутами. Я рассмотрела работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

Список литературы

1. Кулябов Д. С., Королькова А. В., Геворкян М. Н Лабораторная работа №5 [Электронный ресурс]. RUDN, 2023. URL: https://esystem.rudn.ru/pluginfile.php/2090208/mod_resource/content/2/005-lab_discret_sticky.pdf.