

Лабораторная работа №5

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Федорина Эрнест Васильевич

Российский университет дружбы народов, Москва, Россия

Информация

- Федорина Эрнест Васильевич
- студент
- Российский университет дружбы народов
- 1032216454@pfur.ru
- <https://evfedorina.github.io/ru/>



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.

`chmod` (от англ. *change mode*) — команда для изменения прав доступа к файлам и каталогам, используемая в Unix-подобных операционных системах. Входит в стандарт POSIX, в Coreutils.

Выполнение лабораторной работы

Для начала мы проверим наличие gcc и сделаем так, чтобы система защиты SELinux не мешала выполнению работы (рис. (fig:001?))

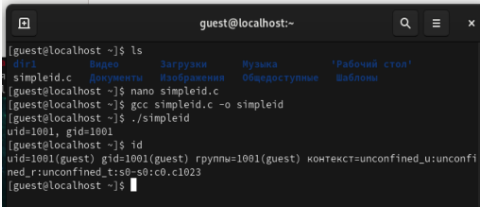
```
[root@localhost guest]# yum install gcc
Обновление репозитория службы управления подписками.
Невозможно прочитать идентификатор клиента

This system is not registered with an entitlement server. You can use "rhc" or "
subscription-manager" to register.

Последняя проверка окончания срока действия метаданных: 0:06:58 назад, Сб 28 сен
2024 22:10:19.
Пакет gcc-11.5.0-2.el9.aarch64 уже установлен.
Зависимости разрешены.
Отсутствуют действия для выполнения.
Выполнено!
[root@localhost guest]# setenforce 0
[root@localhost guest]# getenforce
Permissive
```

Рис. 1: проверка наличия gcc в системе

Создадим файл simpleid.c, напишем код в нём, а далее скомпилируем и запустим, сравним с выполнением команды id. Наш файл работает идентично и выдаёт ID пользователя (рис. (fig:002?))



```
guest@localhost:~  
[guest@localhost ~]$ ls  
dir1      Видео      Загрузки    Музыка      'Рабочий стол'  
simpleid.c Документы  Изображения Общедоступные Шаблоны  
[guest@localhost ~]$ nano simpleid.c  
[guest@localhost ~]$ gcc simpleid.c -o simpleid  
[guest@localhost ~]$ ./simpleid  
uid=1001, gid=1001  
[guest@localhost ~]$ id  
uid=1001(guest) gid=1001(guest) rpnny=1001(guest) контекст=unconfined_u:unconfi  
ned_r:unconfined_t:s0-s0:c0.c1023  
[guest@localhost ~]$
```

Рис. 2: работа с simpleid.c

Изменим код, добавив вывод дополнительных идентификаторов и запустим программу (рис. (fig:003?), (fig:004?))



```
GNU nano 5.6.1 simpleid.c Изменён
#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 ();

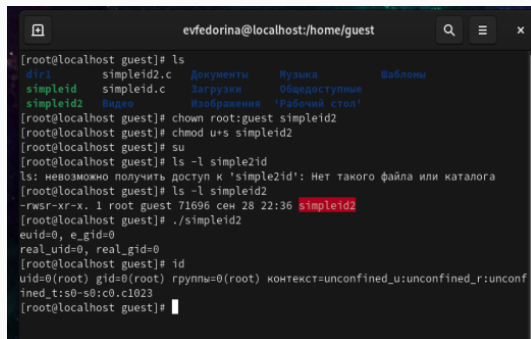
    printf ("euid=%d, e_gid=%d\n", e_uid, e_gid);
    printf ("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
    return 0;
}
```

Рис. 3: код в simpleid2.c

```
[guest@localhost ~]$ gcc simpleid2.c -o simpleid2
[guest@localhost ~]$ ./simpleid2
bash: ./simpleid2: Нет такого файла или каталога
[guest@localhost ~]$ ./simpleid2
euid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@localhost ~]$
```

Рис. 4: запуск simpleid2

Выполним несколько команд от имени суперпользователя, проверим правильность установки новых атрибутов и смены владельца файла, а также ещё раз запустим файл и сравним его работу с командой `id`. Выдают идентичные результаты (рис. (fig:005?))



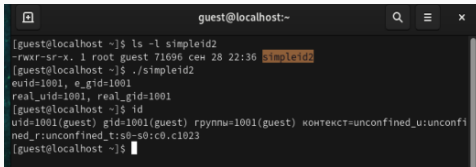
```
evfedorina@localhost:/home/guest

[root@localhost guest]# ls
dir1      simpleid2.c  Документы    Музыка       Шаблоны
simpleid   simpleid.c   Загрузки     Общедоступные
simpleid2  Видео       Изображения  'Рабочий стол'

[root@localhost guest]# chown root:guest simpleid2
[root@localhost guest]# chmod u+s simpleid2
[root@localhost guest]# su
[root@localhost guest]# ls -l simple2id
ls: невозможно получить доступ к 'simple2id': Нет такого файла или каталога
[root@localhost guest]# ls -l simpleid2
-rwsr-xr-x. 1 root guest 71696 сен 28 22:36 simpleid2
[root@localhost guest]# ./simpleid2
euid=0, e_gid=0
real_uid=0, real_gid=0
[root@localhost guest]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@localhost guest]#
```

Рис. 5: работа с атрибутами файла simpleid2

Проделаем то же самое с помощью SetGID-бита (рис. (fig:006?))

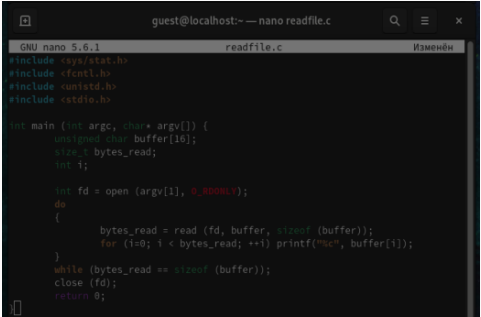
A terminal window titled 'guest@localhost:~' with search, menu, and close icons. It shows the following commands and output:

```
[guest@localhost ~]$ ls -l simpleid2
-rwxr-sr-x. 1 root guest 71696 сен 28 22:36 simpleid2
[guest@localhost ~]$ ./simpleid2
euid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@localhost ~]$ id
uid=1001(guest) gid=1001(guest) rpyнны=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@localhost ~]$
```

The terminal window shows the execution of the simpleid2.c program. The file simpleid2 is listed with permissions -rwxr-sr-x, owned by root and group guest. Running the program outputs the effective user ID (euid) and effective group ID (e_gid) as 1001. The 'id' command confirms the user is guest (uid=1001) and the group is guest (gid=1001). The context is unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023.

Рис. 6: повторение действий с помощью SetGID

Напишем код для файла readfile.c (рис. (fig:007?))



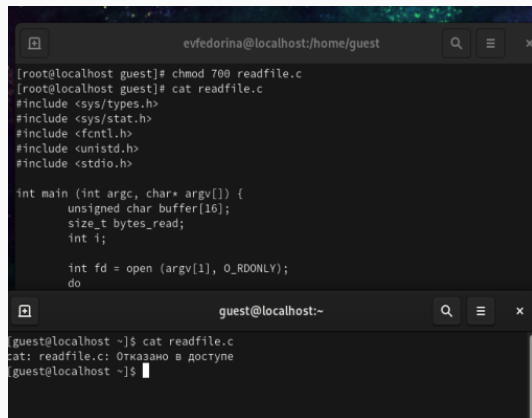
```
GNU nano 5.6.1      readfile.c      Изменён
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.h>

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. 7: код readfile.c

Сменим владельца, чтобы прочитать файл мог только суперпользователь. Попробуем прочитать файл от имени guest. Отказывает в доступе. Далее сменим владельца и установим SetUID-бит, проверим чтение файла, получилось (рис. (fig:008?))



The image shows two terminal windows. The top window is titled 'evfedorina@localhost:/home/guest' and shows a root user performing two commands: 'chmod 700 readfile.c' and 'cat readfile.c'. The output of the cat command shows the source code of 'readfile.c', which includes headers for `<sys/types.h>`, `<sys/stat.h>`, `<fcntl.h>`, `<unistd.h>`, and `<stdio.h>`. The main function declares a buffer, bytes_read, and i, then attempts to open a file specified by argv[1] in read-only mode. The bottom window is titled 'guest@localhost:~' and shows a user attempting to run 'cat readfile.c', which results in the error message 'cat: readfile.c: Отказано в доступе' (Access denied).

```
evfedorina@localhost:/home/guest
[root@localhost guest]# chmod 700 readfile.c
[root@localhost guest]# cat readfile.c
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdio.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
```

```
guest@localhost:~
guest@localhost ~]$ cat readfile.c
cat: readfile.c: Отказано в доступе
guest@localhost ~]$
```

Рис. 8: работа с доступами и владельцами readfile.c

Исследуем Sticky-бит (рис. (fig:009?))

```

[root@localhost ~]# cd tmp
[root@localhost tmp]# cd ..
[root@localhost ~]# cd /tmp
[root@localhost ~]# chmod -t /tmp
[root@localhost ~]# cd /tmp
[root@localhost ~]#

[guest2@localhost ~]$ echo "test1" > file01.txt
bash: file01.txt: Ошибка в доступе
[guest2@localhost tmp]$ echo "test2" >> file01.txt
bash: file01.txt: Ошибка в доступе
[guest2@localhost tmp]$ echo "test3" >> file01.txt
bash: file01.txt: Ошибка в доступе
[guest2@localhost tmp]$ rm file01.txt
rm: удалить защищенный или записи пустой обычный файл 'file01.txt'? y
rm: невозможно удалить 'file01.txt': Ошибка в доступе
[guest2@localhost tmp]$ su guest
Попытка:

Последний вход в систему: 65 сен 28 22:39:56 MSK 2024 на pts/0
[root@localhost ~]# cd guest
-bash: cd: guest: Нет такого файла или каталога
[root@localhost ~]# cd tmp
-bash: cd: tmp: Нет такого файла или каталога
[root@localhost ~]# ls
anaconda-ks.cfg
[root@localhost ~]# cd ~
[root@localhost ~]# ls
anaconda-ks.cfg
[root@localhost ~]# cd ~
[root@localhost ~]# ls
anaconda-ks.cfg
[root@localhost ~]# cd ~
[root@localhost ~]# ls
anaconda-ks.cfg
[root@localhost ~]# cd home
-bash: cd: home: Нет такого файла или каталога
[root@localhost ~]# cd guest
-bash: cd: guest: Нет такого файла или каталога
[root@localhost ~]# su guest
[guest@localhost tmp]$ cd ..
[guest@localhost tmp]$ cd /tmp
bash: cd: guest: Нет такого файла или каталога
[guest@localhost tmp]$ ls
file01.txt  etc  lib  media  opt  root  sbin  sys  usr
bin  boot  home  lib64  root  proc  run  srv  var
[guest@localhost tmp]$ cd /tmp
[guest@localhost tmp]$ cd home
[guest@localhost home]$ ls
efedorina guest guest2
[guest@localhost home]$ cd guest
[guest@localhost ~]# cd tmp
[guest@localhost tmp]$ su guest2
Попытка:

[guest2@localhost tmp]$ ls -l / | grep tmp
drwxrwxr-x. 15 root root 4096 сен 28 23:08 tmp
[guest2@localhost tmp]$ cat file01.txt
test1
test2
test3
[guest2@localhost tmp]$ echo "test2" >> file01.txt
bash: file01.txt: Ошибка в доступе
[guest2@localhost tmp]$ echo "test3" >> file01.txt
bash: file01.txt: Ошибка в доступе
[guest2@localhost tmp]$
```

Рис. 9: работа со Sticky битом

Выводы

Изучил механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. 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.

Список литературы:

1. Chmod[Электронный ресурс] - <https://en.wikipedia.org/wiki/Chmod>