

# Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

---

Норсоян Шушаник Гагиковна

11 марта, 2024, Москва, Россия

Российский Университет Дружбы Народов

# Цели и задачи

---

- SUID - разрешение на установку идентификатора пользователя. Это бит разрешения, который позволяет пользователю запускать исполняемый файл с правами владельца этого файла.
- SGID - разрешение на установку идентификатора группы. Принцип работы очень похож на SUID с отличием, что файл будет запускаться пользователем от имени группы, которая владеет файлом.

## Цель лабораторной 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.

# **Выполнение лабораторной работы**

---

# Программа simpleid

```
[guest@sgnorsoyan ~]$  
[guest@sgnorsoyan ~]$ cd  
[guest@sgnorsoyan ~]$ mkdir lab5  
[guest@sgnorsoyan ~]$ cd lab5  
[guest@sgnorsoyan lab5]$ touch simpleid.c  
[guest@sgnorsoyan lab5]$ gedit simpleid.c  
[guest@sgnorsoyan lab5]$ gcc simpleid.c  
[guest@sgnorsoyan lab5]$ gcc simpleid.c -o simpleid  
[guest@sgnorsoyan lab5]$ ./simpleid  
uid=1001, gid=1001  
[guest@sgnorsoyan lab5]$ id  
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_  
[guest@sgnorsoyan lab5]$
```

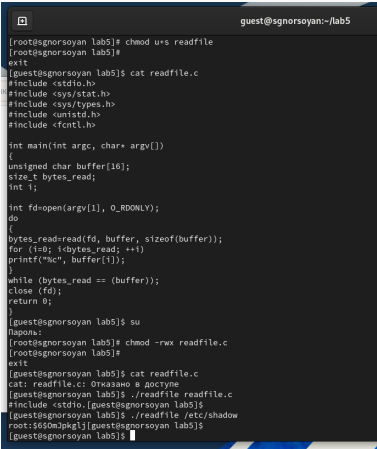
Figure 1: результат программы simpleid

# Программа simpleid2

```
[guest@sgnorsoyan lab5]$  
[guest@sgnorsoyan lab5]$ touch simpleid2.c  
[guest@sgnorsoyan lab5]$ gedit simpleid2.c  
[guest@sgnorsoyan lab5]$ gcc simpleid2.c  
[guest@sgnorsoyan lab5]$ gcc simpleid2.c -o simpleid2  
[guest@sgnorsoyan lab5]$ ./simpleid2  
e_uid=1001, e_gid=1001  
real_uid=1001, real_gid=1001  
[guest@sgnorsoyan lab5]$ su  
Пароль:  
[root@sgnorsoyan lab5]# chown root:guest simpleid2  
[root@sgnorsoyan lab5]# chmod u+s simpleid2  
[root@sgnorsoyan lab5]# ./simpleid2  
e_uid=0, e_gid=0  
real_uid=0, real_gid=0  
[root@sgnorsoyan lab5]# id  
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:sudo_t:s0  
[root@sgnorsoyan lab5]# chmod g+s simpleid2  
[root@sgnorsoyan lab5]# ./simpleid2  
e_uid=0, e_gid=1001  
real_uid=0, real_gid=0  
[root@sgnorsoyan lab5]#  
exit  
[guest@sgnorsoyan lab5]$ ./simpleid2  
e_uid=0, e_gid=1001  
real_uid=1001, real_gid=1001  
[guest@sgnorsoyan lab5]$
```

Figure 2: результат программы simpleid2

# Программа readfile

A terminal window with a dark background and light-colored text. The prompt is 'guest@sgnorsoyan:~/lab5'. The user enters 'chmod u+s readfile'. The prompt changes to '[root@sgnorsoyan lab5]#'. The user enters 'exit'. The prompt changes back to 'guest@sgnorsoyan lab5\$'. The user enters 'cat readfile.c'. The terminal displays the contents of the file 'readfile.c', which includes headers for <stdio.h>, <sys/stat.h>, <sys/types.h>, <unistd.h>, and <fcntl.h>. The main function opens a file specified by argv[1] in read-only mode, reads it into a buffer, and prints the buffer's contents. After displaying the code, the user enters 'su'. The prompt changes to '[root@sgnorsoyan lab5]\$'. The user enters 'chmod -rwx readfile.c'. The prompt changes back to '[root@sgnorsoyan lab5]#'. The user enters 'exit'. The prompt changes back to 'guest@sgnorsoyan lab5\$'. The user enters 'cat: readfile.c: Отказано в доступе'. The user then enters './readfile readfile.c'. The terminal displays the output of the program, which is the contents of 'readfile.c' printed line by line. The user then enters './readfile /etc/shadow'. The terminal displays the contents of the '/etc/shadow' file. The user then enters 'root:\$6\$0mJpkg1j[guest@sgnorsoyan lab5]\$'. The prompt changes to 'root@sgnorsoyan lab5\$'.

```
guest@sgnorsoyan:~/lab5

[root@sgnorsoyan lab5]# chmod u+s readfile
[root@sgnorsoyan lab5]#
exit
[guest@sgnorsoyan lab5]$ cat readfile.c
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <fcntl.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 == (buffer));
    close (fd);
    return 0;
}
[guest@sgnorsoyan lab5]$ su
[root@sgnorsoyan lab5]# chmod -rwx readfile.c
[root@sgnorsoyan lab5]#
exit
[guest@sgnorsoyan lab5]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@sgnorsoyan lab5]$ ./readfile readfile.c
#include <stdio.h>[guest@sgnorsoyan lab5]$
[guest@sgnorsoyan lab5]$ ./readfile /etc/shadow
root:$6$0mJpkg1j[guest@sgnorsoyan lab5]$
[guest@sgnorsoyan lab5]$
```

Figure 3: результат программы readfile



# Исследование Sticky-бита

```
guest@sgnorsoyan lab5]$  
guest@sgnorsoyan lab5]$ echo test >> /tmp/file01.txt  
guest@sgnorsoyan lab5]$ chmod g+rwX /tmp/file01.txt  
guest@sgnorsoyan lab5]$ su guset2  
u: user guset2 does not exist or the user entry does not contain all the required fields  
guest@sgnorsoyan lab5]$ su guest2  
аполь:  
guest2@sgnorsoyan lab5]$ cd /tmp  
guest2@sgnorsoyan tmp]$ cat file01.txt  
est  
guest2@sgnorsoyan tmp]$ echo test2 >> file01.txt  
guest2@sgnorsoyan tmp]$ cat file01.txt  
est  
est2  
guest2@sgnorsoyan tmp]$ echo test3 > file01.txt  
guest2@sgnorsoyan tmp]$ rm file01.txt  
m: невозможно удалить 'file01.txt': Операция не позволена  
guest2@sgnorsoyan tmp]$ su  
аполь:  
root@sgnorsoyan tmp]# chmod -t /tmp  
root@sgnorsoyan tmp]#  
xit  
guest2@sgnorsoyan tmp]$ rm file01.txt  
guest2@sgnorsoyan tmp]$
```

**Figure 4:** исследование Sticky-бита

## **Выводы**

---

Изучили механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получили практические навыки работы в консоли с дополнительными атрибутами. Также мы рассмотрели работу механизма смены идентификатора процессов пользователей и влияние бита Sticky на запись и удаление файлов.