

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Седов Никита НПИбд-02-19

3 октября, 2022, Москва, Россия

Российский Университет Дружбы Народов

Цели и задачи

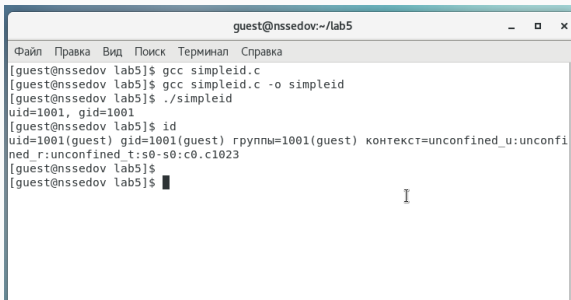
- SUID - разрешение на установку идентификатора пользователя. Это бит разрешения, который позволяет пользователю запускать исполняемый файл с правами владельца этого файла.
- SGID - разрешение на установку идентификатора группы. Принцип работы очень похож на SUID с отличием, что файл будет запускаться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

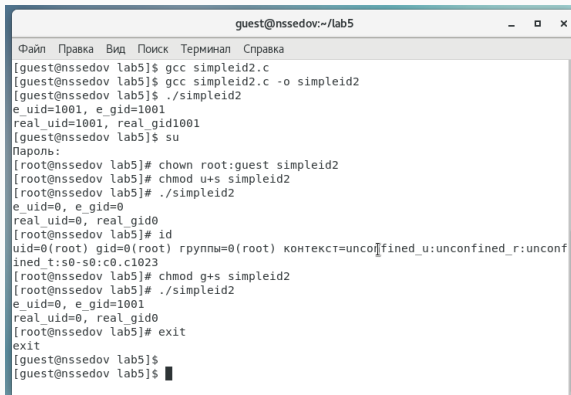
Программа simpleid

A terminal window titled 'guest@nssedov:~/lab5' with standard window controls. The terminal shows the compilation and execution of a program named 'simpleid'. The user runs 'gcc simpleid.c', then 'gcc simpleid.c -o simpleid', and finally './simpleid'. The output of the program is 'uid=1001, gid=1001'. The user then runs 'id', which outputs 'uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023'. The terminal ends with a prompt for the user to enter more commands.

```
guest@nssedov:~/lab5
Файл Правка Вид Поиск Терминал Справка
[guest@nssedov lab5]$ gcc simpleid.c
[guest@nssedov lab5]$ gcc simpleid.c -o simpleid
[guest@nssedov lab5]$ ./simpleid
uid=1001, gid=1001
[guest@nssedov lab5]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfi
ned_r:unconfined_t:s0-s0:c0.c1023
[guest@nssedov lab5]$
[guest@nssedov lab5]$
```

Figure 1: результат программы simpleid

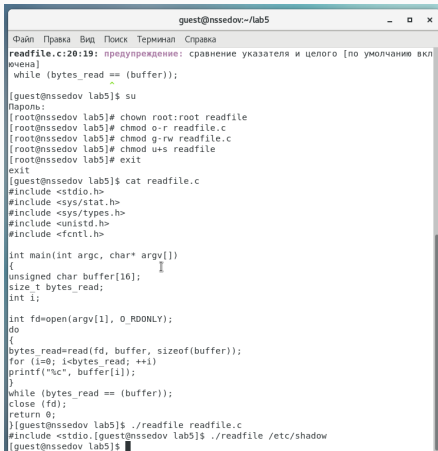
Программа simpleid2



```
guest@nssedov:~/lab5
Файл  Правка  Вид  Поиск  Терминал  Справка
[guest@nssedov lab5]$ gcc simpleid2.c
[guest@nssedov lab5]$ gcc simpleid2.c -o simpleid2
[guest@nssedov lab5]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@nssedov lab5]$ su
Пароль:
[root@nssedov lab5]# chown root:guest simpleid2
[root@nssedov lab5]# chmod u+s simpleid2
[root@nssedov lab5]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@nssedov lab5]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@nssedov lab5]# chmod g+s simpleid2
[root@nssedov lab5]# ./simpleid2
e_uid=0, e_gid=1001
real_uid=0, real_gid=0
[root@nssedov lab5]# exit
exit
[guest@nssedov lab5]$
[guest@nssedov lab5]$ █
```

Figure 2: результат программы simpleid2

Программа readfile

A screenshot of a terminal window titled 'guest@nssedov:~/lab5'. The terminal shows the execution of a C program named 'readfile.c'. The program is compiled and run as root. It opens a file specified by the first command-line argument in read-only mode and prints its contents character by character. The output shows the contents of '/etc/shadow'.

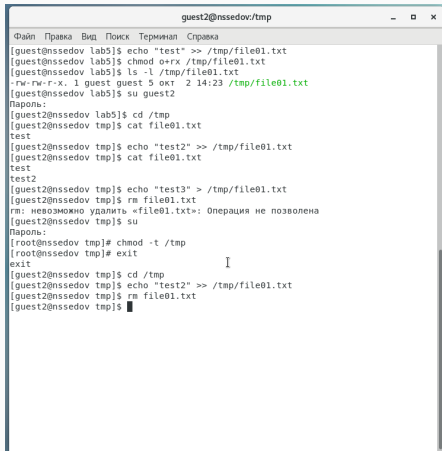
```
guest@nssedov:~/lab5
Файл  Провка Вид  Поиск Терминал Справка
readfile.c:20:19: предупреждение: сравнение указателя и целого [по умолчанию включена]
while (bytes_read == (buffer));
                  ^
[guest@nssedov lab5]$ su
Пароль:
[root@nssedov lab5]# chown root:root readfile
[root@nssedov lab5]# chmod o-r readfile.c
[root@nssedov lab5]# chmod g-rw readfile.c
[root@nssedov lab5]# chmod u+s readfile
[root@nssedov lab5]# exit
exit
[guest@nssedov 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@nssedov lab5]$ ./readfile readfile.c
#include <stdio.h>[guest@nssedov lab5]$ ./readfile /etc/shadow
[guest@nssedov lab5]$
```

Figure 3: результат программы readfile

Исследование Sticky-бита



```
guest2@nssedov/tmp
Файл  Правка  Вид  Поиск  Терминал  Справка
[guest@nssedov lab5]$ echo "test" >> /tmp/file01.txt
[guest@nssedov lab5]$ chmod o+rx /tmp/file01.txt
[guest@nssedov lab5]$ ls -l /tmp/file01.txt
-rw-rw-r-x. 1 guest guest 5 окт  2 14:23 /tmp/file01.txt
[guest@nssedov lab5]$ su guest2
Пароль:
[guest2@nssedov lab5]$ cd /tmp
[guest2@nssedov tmp]$ cat file01.txt
test
[guest2@nssedov tmp]$ echo "test2" >> /tmp/file01.txt
[guest2@nssedov tmp]$ cat file01.txt
test
test2
[guest2@nssedov tmp]$ echo "test3" > /tmp/file01.txt
[guest2@nssedov tmp]$ rm file01.txt
rm: невозможно удалить «file01.txt»: Операция не позволена
[guest2@nssedov tmp]$ su
Пароль:
[root@nssedov tmp]# chmod -t /tmp
[root@nssedov tmp]# exit
exit
[guest2@nssedov tmp]$ cd /tmp
[guest2@nssedov tmp]$ echo "test2" >> /tmp/file01.txt
[guest2@nssedov tmp]$ rm file01.txt
[guest2@nssedov 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.