

Отчёт по лабораторной работе №5

**Дискреционное разграничение прав в Linux. Исследование влияния
дополнительных атрибутов**

Косолапов Степан Эдуардович НПИбд-01-20

Содержание

1	Цель работы	5
2	Выполнение работы	6
3	Выводы	14

Список иллюстраций

figno:1 simpleid.c	6
figno:2 Компиляция simpleid.c	6
figno:3 Сравнение вывода simpleid.c и id	7
figno:4 simpleid2.c	7
figno:5 Компиляция и запуск simpleid2.c	7
figno:6 Устанавливаем владельца файла simpleid2 и добавляем setuid бит	8
figno:7 Выполняем simpleid2 с битом setuid	8
figno:8 readfile.c	9
figno:9 Установка прав на файл readfile.c только для root	9
figno:10 Установка setuid бита на файл readfile	10
figno:11 Чтение readfile.c с помощью readfile с установленным setuid битом	10
figno:12 Чтение /etc/shadow с помощью readfile с установленным setuid битом . .	11
figno:13 Создание файла в sticky директории	11
figno:14 Операции над файлом в sticky директории	12
figno:15 Удаление атрибута sticky	12
figno:16 Операции над файлом в директории без sticky	13

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 Выполнение работы

1. Входим в систему от имени пользователя guest. Создаем программу simpleid.c:

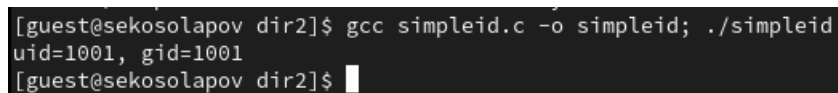


```
guest@sekosolapov:~/dir2
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

int
main()
{
    uid_t uid = geteuid();
    gid_t gid = getegid();
    printf("uid=%d, gid=%d\n", uid, gid);
    return 0;
}
```

simpleid.c

2. Компилируем программу и убеждаемся, что файл программы создан.



```
[guest@sekosolapov dir2]$ gcc simpleid.c -o simpleid; ./simpleid
uid=1001, gid=1001
[guest@sekosolapov dir2]$
```

компиляция simpleid.c

3. Выполняем программу simpleid и выполняем системную программу id. Видим, что айдишники показываются такие же, но id показывает более расширенную информацию.

```
[guest@sekosolapov dir2]$ ./simpleid
uid=1001, gid=1001
[guest@sekosolapov dir2]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@sekosolapov dir2]$
```

сравнение вывода simpleid.c и id

4. Усложняем программу, добавив вывод действительных идентификаторов. Получившуюся программу называем simpleid2.c.

```
guest@sekosolapov:~/dir2
#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;
}
```

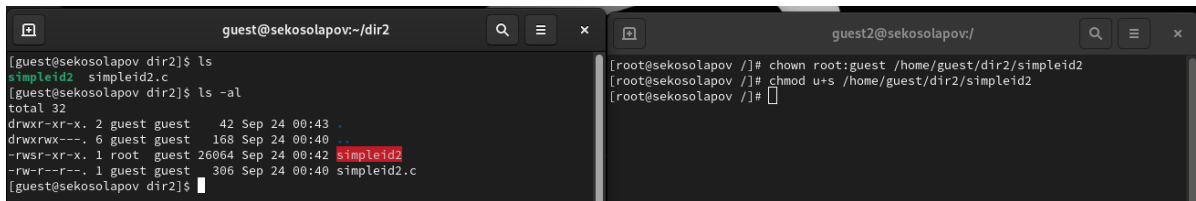
simpleid2.c

5. Компилируем и запускаем simpleid2.c.

```
guest@sekosolapov:~/dir2
[guest@sekosolapov dir2]$ gcc simpleid2.c -o simpleid2; ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@sekosolapov dir2]$ S
```

компиляция и запуск simpleid2.c

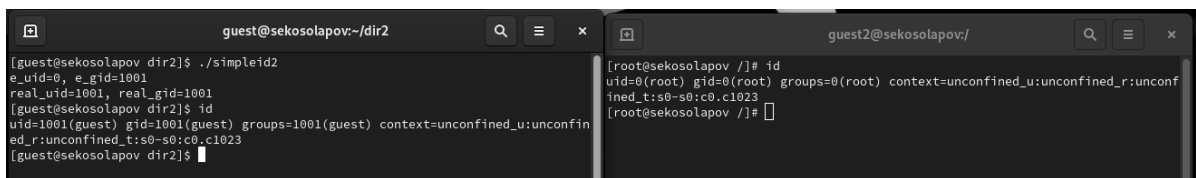
6. От имени суперпользователя выполняем команды: `chown root:guest /home/guest/simpleid2`
`chmod u+s /home/guest/simpleid2`
7. Выполняем проверку правильности установки новых атрибутов и смены владельца файла `simpleid2`



The image shows two terminal windows. The left window, titled 'guest@sekosolapov:~/dir2', shows the command `ls` output for `simpleid2` and `simpleid2.c`. The file `simpleid2` is owned by root:guest and has permissions `-rwsr-xr-x`. The right window, titled 'guest2@sekosolapov:/', shows the commands `chown root:guest /home/guest/dir2/simpleid2` and `chmod u+s /home/guest/dir2/simpleid2` being executed successfully.

устанавливаем владельца файла `simpleid2` и добавляем `setuid` бит

8. Запускаем программы `simpleid2` и `id`. Видим, что при выполнении файла `simpleid2` он берет `id` владельца.



The image shows two terminal windows. The left window, titled 'guest@sekosolapov:~/dir2', shows the command `./simpleid2` being executed, which outputs `e_uid=0, e_gid=1001` and `real_uid=1001, real_gid=1001`. The right window, titled 'guest2@sekosolapov:/', shows the command `id` being executed, which outputs `uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023`.

выполняем `simpleid2` с битом `setuid`

9. Создаём программу `readfile.c`:


```
guest@sekosolapov:~/dir2

#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;
}

~
~
~
"readfile.c" 28L, 427B                                28,1      All
```

readfile.c

10. Компилируем ее командой: `gcc readfile.c -o readfile`
11. Меняем владельца у файла `readfile.c` на `root` и выделяем права на чтение только для владельца. Проверяем, что пользователь `guest` не может прочитать файл `readfile.c`.

```
guest@sekosolapov:~/dir2
[guest@sekosolapov dir2]$ gcc readfile.c -o readfile
[guest@sekosolapov dir2]$ ls -al | grep readfile.c
-rw-r--r--. 1 guest guest 427 Sep 24 00:55 readfile.c
[guest@sekosolapov dir2]$ ls -al | grep readfile.c
-rw-r--r--. 1 root  guest 427 Sep 24 00:55 readfile.c
[guest@sekosolapov dir2]$ ls -al | grep readfile.c
-rwx-----. 1 root  guest 427 Sep 24 00:55 readfile.c
[guest@sekosolapov dir2]$ cat readfile.c
cat: readfile.c: Permission denied
[guest@sekosolapov dir2]$

guest2@sekosolapov:/
[root@sekosolapov /]# chown root /home/guest/dir2/readfile.c
[root@sekosolapov /]# chmod 700 /home/guest/dir2/readfile.c
[root@sekosolapov /]#
```

установка прав на файл `readfile.c` только для `root`

12. Меняем у программы `readfile` владельца на `root` и устанавливаем SetUID-бит.

```
guest2@sekosolapov:/home/guest/dir2

[root@sekosolapov dir2]# chown root readfile
[root@sekosolapov dir2]# ls -al | grep readfile
-rwxr-xr-x. 1 root  guest 26008 Sep 24 00:55 readfile
-rwx-----. 1 root  guest  427 Sep 24 00:55 readfile.c
[root@sekosolapov dir2]# chmod u+s readfile
[root@sekosolapov dir2]# ls -al | grep readfile
-rwsr-xr-x. 1 root  guest 26008 Sep 24 00:55 readfile
-rwx-----. 1 root  guest  427 Sep 24 00:55 readfile.c
[root@sekosolapov dir2]#
```

установка setuid бита на файл readfile

13. Проверяем, может ли программа readfile прочитать файл readfile.c. Видим, что программа может прочитать файл, потому что установлен SetUID бит, который позволяет программе при запуске получать права своего владельца, которым установлен root.

```
guest@sekosolapov:~/dir2
[guest@sekosolapov dir2]$ ./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;

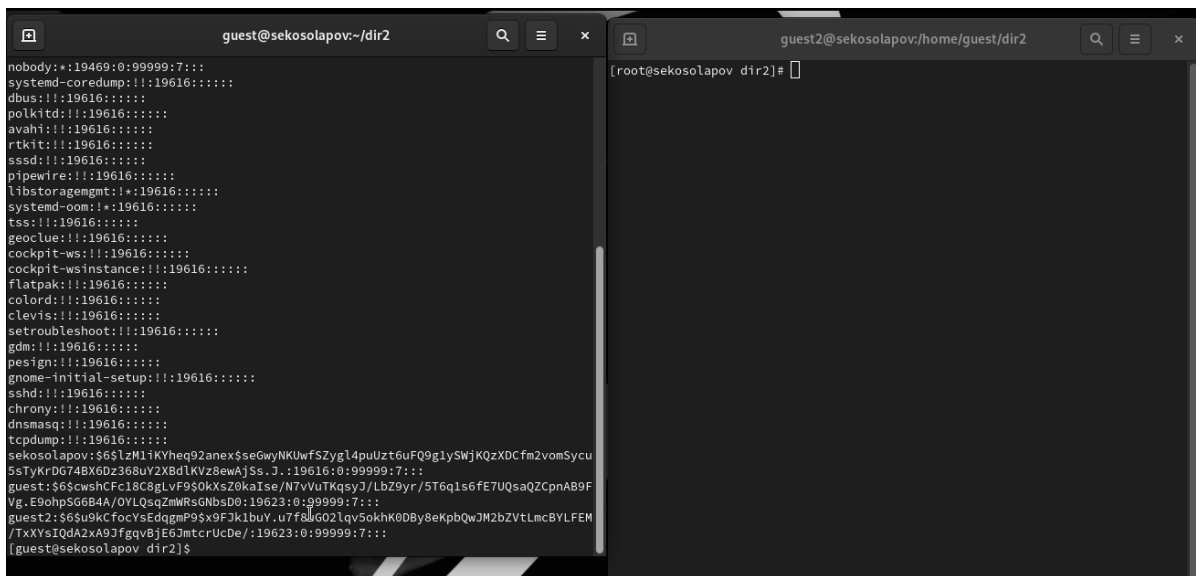
    int fd = open(argv[1], O_RDONLY);
    do
    {
        bytes_read = read(fd, buffer, sizeof (buffer));
        for (i = 0; i < bytes_read; ++i)
        {
            printf("%c", buffer[i]);
        }
    }
    while (bytes_read == sizeof(buffer));
    close(fd);

    return 0;
}
[guest@sekosolapov dir2]$

guest2@sekosolapov:/home/guest/dir2
[root@sekosolapov dir2]# chown root readfile
[root@sekosolapov dir2]# ls -al | grep readfile
-rwxr-xr-x. 1 root  guest 26008 Sep 24 00:55 readfile
-rwx-----. 1 root  guest  427 Sep 24 00:55 readfile.c
[root@sekosolapov dir2]# chmod u+s readfile
[root@sekosolapov dir2]# ls -al | grep readfile
-rwsr-xr-x. 1 root  guest 26008 Sep 24 00:55 readfile
-rwx-----. 1 root  guest  427 Sep 24 00:55 readfile.c
[root@sekosolapov dir2]#
```

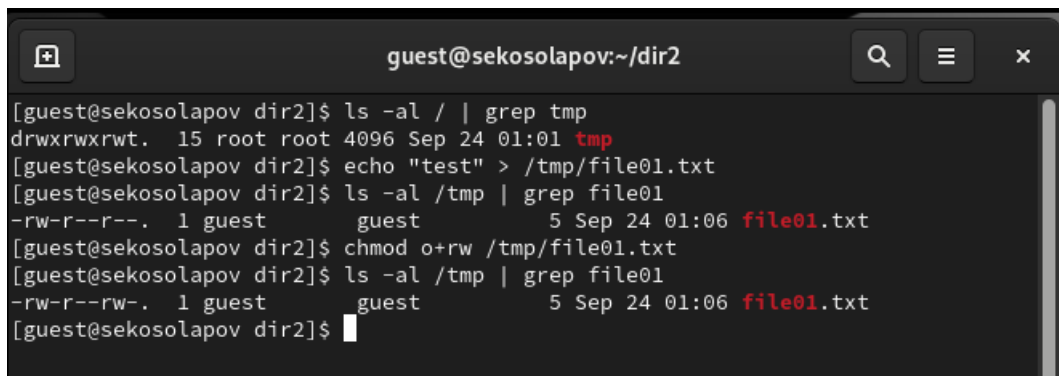
чтение readfile.c с помощью readfile с установленным setuid битом

14. Проверяем, может ли программа readfile прочитать файл /etc/shadow. Ситуация аналогичная предыдущему пункту. Мы можем прочитать этот файл, потому что у root есть к нему доступ на чтение.



чтение /etc/shadow с помощью readfile с установленным setuid битом

15. Выясняем, установлен ли атрибут Sticky на директории /tmp. Видим, что атрибут установлен (атрибут t). От имени пользователя guest создаём file01.txt в директории /tmp со словом test. Смотрим атрибуты у только что созданного файла и разрешаем чтение и запись для категории пользователей «все остальные».

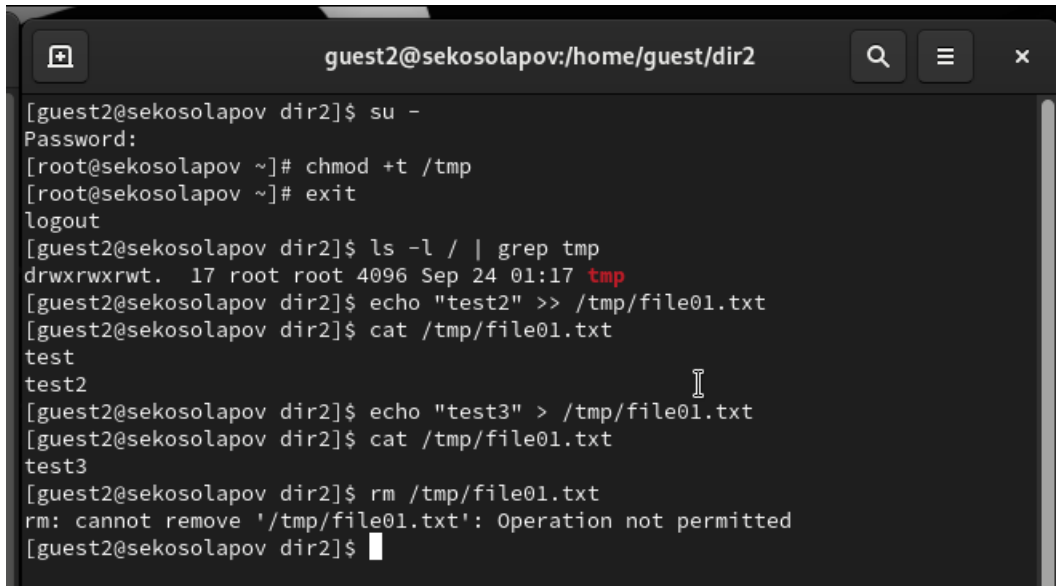


создание файла в sticky директории

На самом деле, тут у пользователя guest2 не будет прав, потому что он находится в группе владельца файла и нам нужно добавить права на чтение для группы, а не для всех остальных.

```
chmod g+rw /tmp/file01.txt
```

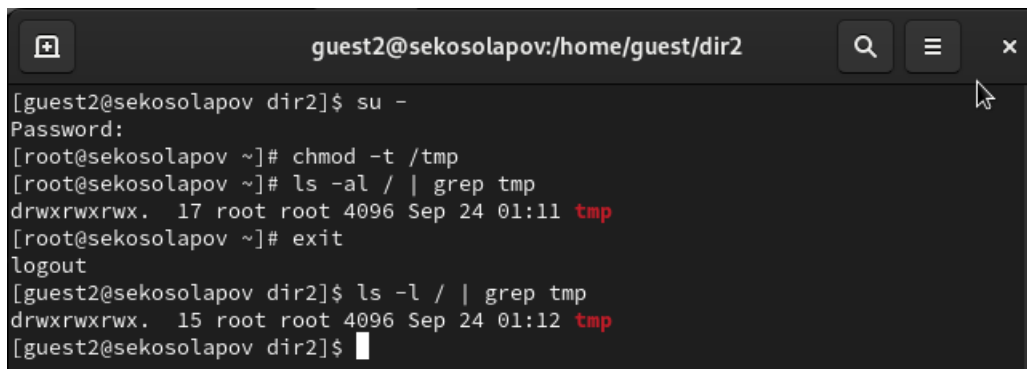
16. От пользователя guest2 пробуем выполнить привычные операции с файлом /tmp/file01.txt. У нас получилось выполнить - чтение, дозапись, перезапись. Не получилось сделать удаление файла. Как раз из-за установленного на директории tmp атрибута tmp.



```
guest2@sekosolapov:/home/guest/dir2
[guest2@sekosolapov dir2]$ su -
Password:
[root@sekosolapov ~]# chmod +t /tmp
[root@sekosolapov ~]# exit
logout
[guest2@sekosolapov dir2]$ ls -l / | grep tmp
drwxrwxrwt. 17 root root 4096 Sep 24 01:17 tmp
[guest2@sekosolapov dir2]$ echo "test2" >> /tmp/file01.txt
[guest2@sekosolapov dir2]$ cat /tmp/file01.txt
test
test2
[guest2@sekosolapov dir2]$ echo "test3" > /tmp/file01.txt
[guest2@sekosolapov dir2]$ cat /tmp/file01.txt
test3
[guest2@sekosolapov dir2]$ rm /tmp/file01.txt
rm: cannot remove '/tmp/file01.txt': Operation not permitted
[guest2@sekosolapov dir2]$
```

операции над файлом в sticky директории

17. От имени суперпользователя убираем атрибут sticky.



```
guest2@sekosolapov:/home/guest/dir2
[guest2@sekosolapov dir2]$ su -
Password:
[root@sekosolapov ~]# chmod -t /tmp
[root@sekosolapov ~]# ls -al / | grep tmp
drwxrwxrwx. 17 root root 4096 Sep 24 01:11 tmp
[root@sekosolapov ~]# exit
logout
[guest2@sekosolapov dir2]$ ls -l / | grep tmp
drwxrwxrwx. 15 root root 4096 Sep 24 01:12 tmp
[guest2@sekosolapov dir2]$
```

снятие атрибута sticky

18. Пробуем повторить операции без атрибута sticky у директории tmp. Теперь мы можем выполнять все операции, которые разрешены для группы владельца файла, в нашем случае это дозапись, перезапись, чтение и удаление файла.

```
guest@sekosolapov:~/dir2
[guest@sekosolapov dir2]$ ls -al / | grep tmp
drwxrwxrwt. 15 root root 4096 Sep 24 01:01 tmp
[guest@sekosolapov dir2]$ echo "test" > /tmp/file01.txt
[guest@sekosolapov dir2]$ ls -al /tmp | grep file01
-rw-r--r--. 1 guest guest 5 Sep 24 01:06 file01.txt
[guest@sekosolapov dir2]$ chmod o+rw /tmp/file01.txt
[guest@sekosolapov dir2]$ ls -al /tmp | grep file01
-rw-r--rw-. 1 guest guest 5 Sep 24 01:06 file01.txt
[guest@sekosolapov dir2]$ ls -al /tmp | grep file01
-rw-r--rw-. 1 guest guest 5 Sep 24 01:06 file01.txt
[guest@sekosolapov dir2]$ chmod g+rw /tmp/file01.txt
[guest@sekosolapov dir2]$ ls -al /tmp | grep file01
-rw-rw-rw-. 1 guest guest 5 Sep 24 01:06 file01.txt
[guest@sekosolapov dir2]$

guest2@sekosolapov:/home/guest/dir2
[guest2@sekosolapov dir2]$ cat /tmp/file01.txt
test
[guest2@sekosolapov dir2]$ echo "test2" >> /tmp/file01.txt
[guest2@sekosolapov dir2]$ cat /tmp/file01.txt
test
test2
[guest2@sekosolapov dir2]$ echo "test3" > /tmp/file01.txt
[guest2@sekosolapov dir2]$ cat /tmp/file01.txt
test3
[guest2@sekosolapov dir2]$ rm /tmp/file01.txt
[guest2@sekosolapov dir2]$ ls -l /tmp | grep file01
[guest2@sekosolapov dir2]$
```

операции над файлом в директории без sticky

19. В конце возвращаем атрибут sticky директории tmp.

3 Выводы

В данной работе мы изучили атрибуты sticky, и биты setgid и setuid и их влияние на различные аспекты работы системы. Атрибут sticky, применяемый к директориям, позволяет ограничить доступ к файлам в них, предотвращая удаление или переименование файлов другими пользователями. Бит setgid позволяет устанавливать группу-владельца для новых файлов, созданных в директории, что может быть полезно для обеспечения совместного доступа к файлам между пользователями. Бит setuid, с другой стороны, позволяет запускать исполняемые файлы с привилегиями владельца файла, что может быть полезно для выполнения определенных задач, требующих повышенных привилегий.