Презентация Лабораторной работы №5

По дисциплине Информационная безопасность

Прокошев Н.Е.

07 октября 2023

Информация

Докладчик

- Прокошев Никита Евгеньевич
- студент НФИбд-02-20
- Факультет Физико-Математических и Естественных наук
- Российский университет дружбы народов
- 1032202460@rudn.ru
- https://github.com/neprokoshev

Вводная часть

Цели и задачи

Цель: 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

1. Входим в систему от имени пользователя guest и создаём программу simpleid.c (@pic:001, @pic:002).

```
[nikitaprokoshev@neprokoshev ~]$ su - guest
Password:
[guest@neprokoshev ~]$ touch simpleid.c
```

```c
#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;
}
```

2. Скомплилируем и выполним программу simpleid.c и системную программу id и сравним полученные результаты (@pic:003).

```
[guest@neprokoshev ~]$ gcc simpleid.c -o simpleid
[guest@neprokoshev ~]$ ./simpleid
uid=1001,gid=1001
[guest@neprokoshev ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 1: Компилируем и выполняем программу simpleid.c.

3. Усложним программу, добавив вывод действительных идентификаторов и назовём её simpleid2.c (@pic:004, @pic:005).

```
[guest@neprokoshev ~]$ touch simpleid2.c
```

```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;
}
```

4. Скомпилируем и запустим simpleid2.c (@pic:006).

```
[guest@neprokoshev ~]$ gcc simpleid2.c -o simpleid2
[guest@neprokoshev ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@neprokoshev ~]$ id
uid=1001(guest) gid=1001(guest) groups=1001(guest) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 2: Компилируем и выполняем программу simpleid2.c.

5. От имени суперпользователя выполним команды и выполним проверку правильности установки новых атрибутов и смены владельца файла simpleid2.c (@pic:007).
6. Запустим simpleid2.c и id и сравним результаты (@pic:008).

```
[guest@neprokoshev ~]$ su
Password:
[root@neprokoshev guest]# chown root:guest /home/guest/simpleid2
[root@neprokoshev guest]# chmod u+s /home/guest/simpleid2
[root@neprokoshev guest]# ls -l simpleid2
-rwsr-xr-x. 1 root guest 26064 Oct  5 23:55 simpleid2
```

Рис. 3: Выполняем команды и проверяем правильность установки атрибутов.

```
[root@neprokoshev guest]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@neprokoshev guest]# id
uid=0(root) gid=0(root) groups=0(root) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 4: Выполняем программу simpleid2.c.

7. Создадим программу readfile.c (@pic:009, @pic:010).

```
[root@neprokoshev guest]# touch readfile.c
```

```c
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.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;
}
```

8. Скомпилируем файл readfile.c (@pic:011).

```
[root@neprokoshev guest]# gcc readfile.c -o readfile
```

Рис. 5: Компилируем программу readfile.c.

9. Сменим владельца у файла readfile.c и изменим права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог. Проверим, что пользователь guest не может прочитать файл readfile.c (@pic:012).

10. Сменим у программы readfile владельца и установим SetUID-бит (@pic:013). Проверим, может ли программа readfile прочитать файлы readfile.c и /etc/shadow? (@pic:014, @pic:015).

```
[root@neprokoshev guest]# chown root /home/guest/readfile.c
[root@neprokoshev guest]# chmod 400 readfile.c
[root@neprokoshev guest]# su guest
[guest@neprokoshev ~]$ cat readfile.c
cat: readfile.c: Permission denied
```

Рис. 6: Меняем владельца файла и его права

```
[guest@neprokoshev ~]$ su
Password:
[root@neprokoshev guest]# chown root /home/guest/readfile
[root@neprokoshev guest]# chmod u+s /home/guest/readfile
```

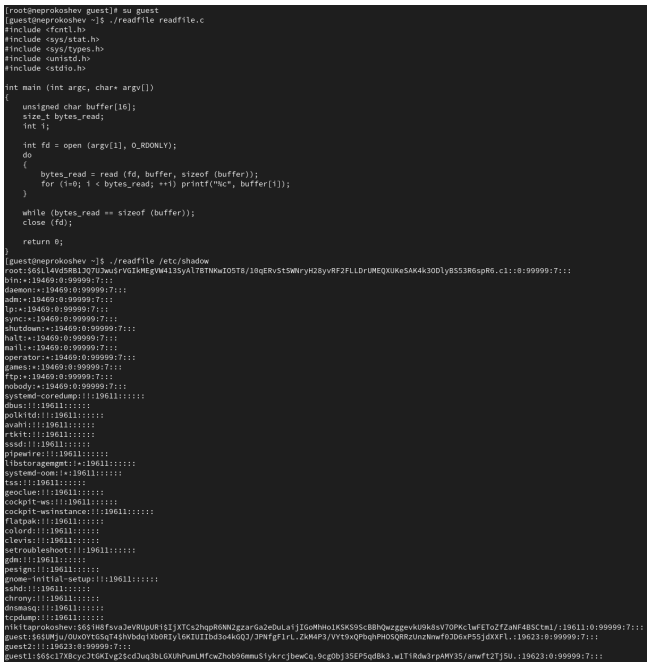
```
[root@neprokoshev guest]# ./readfile readfile.c
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>

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;
}
[root@neprokoshev guest]# ./readfile /etc/shadow
root:$6$lI4VG5RIJQ7U3wu$rVGIkMEgVW4135yAl7BTNKwIO5T8/1OqERvStSWNryH28yvRF2FLLDrUMEQXUKeSAK4k3ODlyBS53R6spR6.c1::0:99999:7:::
bin:*:19469:0:99999:7:::
daemon:*:19469:0:99999:7:::
adm:*:19469:0:99999:7:::
lp:*:19469:0:99999:7:::
sync:*:19469:0:99999:7:::
shutdown:*:19469:0:99999:7:::
halt:*:19469:0:99999:7:::
mail:*:19469:0:99999:7:::
operator:*:19469:0:99999:7:::
games:*:19469:0:99999:7:::
ftp:*:19469:0:99999:7:::
nobody:*:19469:0:99999:7:::
systemd-coredump:!!:19611::::::
dbus:!!:19611::::::
polkitd:!!:19611::::::
avahi:!!:19611::::::
rtkit:!!:19611::::::
sssd:!!:19611::::::
pipewire:!!:19611::::::
libstoragemgmt:!*:19611::::::
systemd-oom:!*:19611::::::
tss:!!:19611::::::
geoclue:!!:19611::::::
cockpit-ws:!!:19611::::::
cockpit-wsinstance:!!:19611::::::
flatpak:!!:19611::::::
colord:!!:19611::::::
clevis:!!:19611::::::
setroubleshoot:!!:19611::::::
gdm:!!:19611::::::
pesign:!!:19611::::::
gnome-initial-setup:!!:19611::::::
sshd:!!:19611::::::
chrony:!!:19611::::::
dnsmasq:!!:19611::::::
tcpdump:!!:19611::::::
nikitaprokoshev:$6$tH8fsvaJeVRUpUR1$IjXTCs2hqpR6NN2gzarGa2eOuLaIjIGoMhHo1KSKS9ScBBhQwzggevkU9k8sV7OPKclwFETOZfZaNF4BSCtm1/:19611:0:99999:7:::
guest:$6$UMju/OUxOYtGSqT4$hVbdqiXbORIylGKIUIIBd3o4kGQ3/JPNfgFlrL.ZkMAP3/VYt9xQPbqhPHOSQRRzUnzNnwf0JD6xP55jdXXfl.:19623:0:99999:7:::
guest2:!!:19623:0:99999:7:::
guest1:$6$clTXBcyCJtGKIvg2$cdJuq3bLGXUhPumLMfcwZhob96mmuSiykrcjbewCq.9cgObj35EP5qdBk3.wlTiRdw3rpAMY35/anwft2TjSU.:19623:0:99999:7:::
```



11. Выясним, установлен ли атрибут Sticky на директории /tmp (@pic:016).

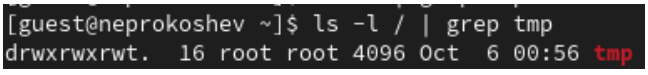


Рис. 7: Проверям Sticky-атрибут.

12. От имени пользователя guest создаём файл file01.txt в директории /tmp со словом “test” (@pic:017).

Рис. 8: Создаём файл file01.txt.

13. Просмотрим атрибуты у только что созданного файла и разрешим чте- ние и запись для категории пользователей «все остальные». От пользователя guest1 попробуем прочитать файл file01.txt (@pic:018).

14. От пользователя guest2 попробуем дозаписать в файл /tmp/file01.txt слово “test2” и проверим содержимое файла (@pic:019).

15. От пользователя guest2 попробуем записать в файл /tmp/file01.txt слово “test3”, стерев при этом всю имеющуюся в файле информацию и проверим содержимое файла (@pic:020).

Рис. 9: Изменяем атрибуты и читаем файл.

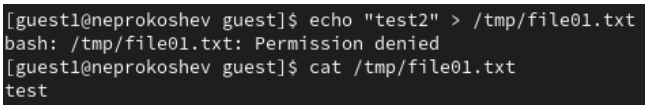


Рис. 10: Дозаписываем слово и проверям файл.

16. От пользователя guest2 попробуем удалить файл /tmp/file01.txt (@pic:021).
17. Повышаем свои права до суперпользователя и выполняем после этого команду, снимающую атрибут t (Sticky-бит) с директории /tmp. После покидаем режим суперпользователя (@pic:022).
18. От пользователя guest2 проверяем, что атрибута t у директории /tmp нет. Повторим предыдущие шаги. Видим, что теперь мы можем удалить этот файл, но не более (@pic:023).
19. Повышаем свои права до суперпользователя и возвращаем атрибут t на ди- ректорию /tmp (@pic:024).

Выводы

В ходе данной лабораторной работы были изучены механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получены практические навыки работы в консоли с дополнительными атрибутами. Рассмотрены работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.

:::

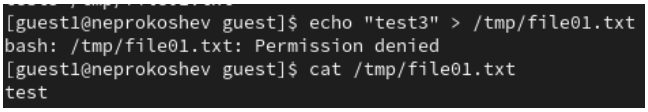


Рис. 11: Дозаписываем слово и проверяем файл.

Рис. 12: Пробуем удалить файл.

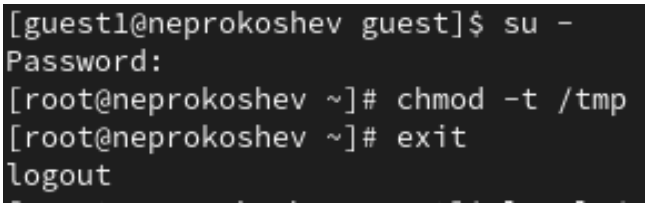


Рис. 13: Снимаем атрибут t.

Рис. 14: Повторяем действия без атрибута t.

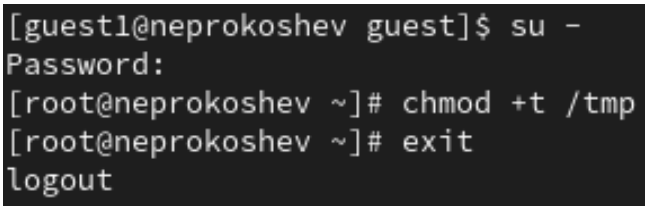


Рис. 15: Возвращаем атрибут t.