Лабораторная работа №5

Дискреционное разграничение прав в Linux. Исследование влияния дополнительных атрибутов

Логинов Егор Игоревич

7 октября 2023

Российский университет дружбы народов, Москва, Россия

Информация

Докладчик

- Логинов Егор Игоревич
- студент НФИбд-01-20
- Российский университет дружбы народов
- 1032201661@pfur.ru
- https://github.com/Y0gu4t

Вводная часть

Актуальность

Логические объекты файловой системы (файлы) являются носителями своеобразных меток, которые привычно называют правами доступа. Некоторые метки действительно означают право выполнения определенного действия пользователя над этим объектом. Важно изучить их для дальнейшего применения на практике.

Объект и предмет исследования

- Атрибуты файлов
- Дистрибутив Rocky
- Дискреционное разграничение доступа

Цели и задачи

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 работы

Создание программы

От имени пользователя guest создадим программу simpleid.c, скомпилируем ее и убедимся, что файл создан. Выполним команды ./simpleid и id и убедимся, что полученные данные совпадают

```
[yegor@yegor ~]$ su guest
Пароль:
[guest@yegor yegor]$ cd /home/guest
[guest@yegor ~]$ nano simpleid.c
[guest@yegor ~]$ gcc simpleid.c -o simpleid
[guest@yegor ~]$ ./simpleid
uid=1001, gid=1001
[guest@yegor ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 1: Использование команд ./simpleid и id

Создание программы

Усложним программу и запишем ее в файл simpleid2.c. Запустим получившуюся программу

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

Рис. 2: Запуск программы simpleid2

От имени суперпользователя установим новые атрибуты и сменим владельца файла simpleid2

```
[root@yegor ~]# chown root:guest /home/guest/simpleid2
[root@yegor ~]# chmod u+s /home/guest/simpleid2
```

Рис. 3: Установки новых атрибутов и смена владельца файла simpleid2

Выполним команду ./simpleid2

```
[root@yegor guest]# ls -l simpleid2
-rwsr-xr-x. 1 root guest 26064 окт  7 20:24 simpleid2
[root@yegor guest]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@yegor guest]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@yegor guest]#
```

Рис. 4: Использование команд ./simpleid2

Создание программы

Проделаем то же самое относительно SetGID-бита

```
[root@yegor guest]# chmod g+s simpleid2
[root@yegor guest]# ls -l simpleid2
-rwsr-sr-x. 1 root guest 26064 окт  7 20:24 simpleid2
[root@yegor guest]# exit
выход
[guest@yegor ~]$ ./simpleid2
e_uid=0, e_gid=1001
real_uid=1001, real_gid=1001
[guest@yegor ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@yegor ~]$
```

Рис. 5: Операции с SetGID-битом

Создадим и скомпилируем программу readfile.c. Сменим владельца у файла readfile.c и изменим права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог

```
[guest@yegor ~]$ nano readfile.c
[guest@yegor ~]$ gcc readfile.c -o readfile
[guest@yegor ~]$ su
Пароль:
[root@yegor guest]# chown root:guest readfile.c
[root@yegor guest]# chmod 700 readfile.c
[root@yegor guest]# exit
exit
[guest@yegor ~]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@yegor ~]$
```

Рис. 6: Изменение владельца и прав файла readfile.c

Сменим у программы readfile владельца и установим SetUID-бит

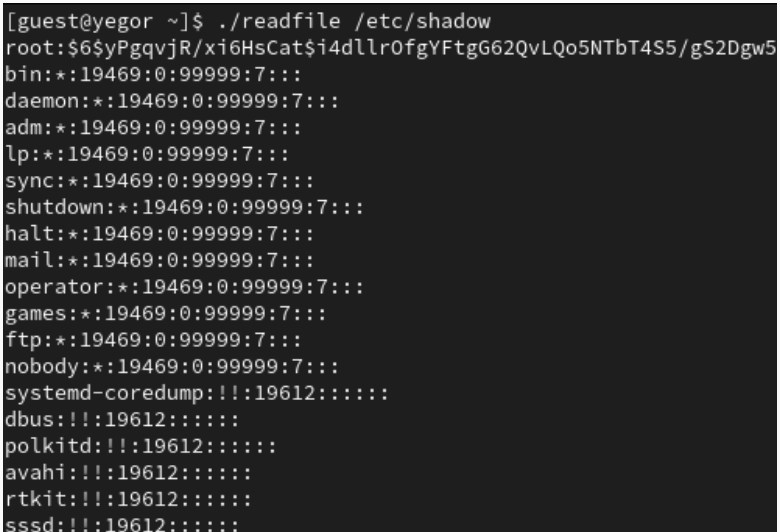
```
[root@yegor guest]# chown root:guest readfile
[root@yegor guest]# chmod u+s readfile
[root@yegor guest]# ls -l readfile
-rwsr-xr-x. 1 root guest 26008 окт  7 20:31 readfile
[root@yegor guest]#
```

Рис. 7: Работа с параметрами readfile

Проверим, может ли программа readfile прочитать файл readfile.c

```
[guest@yegor ~]$ ./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);
```

Проверим, может ли программа readfile прочитать файл /etc/shadow



Выясним, установлен ли атрибут Sticky на директории /tmp

```
[guest@yegor ~]$ ls -l / | grep tmp
drwxrwxrwt.  15 root root 4096 окт  7 20:33 tmp
[guest@yegor ~]$
```

Рис. 10: Чтение атрибутов директории /tmp

От имени пользователя guest создадим файл file01.txt в директории /tmp со словом test. Просмотрим атрибуты у только что созданного файла и разрешим чтение и запись для категории пользователей «все остальные»

```
[guest@yegor ~]$ echo "test" > /tmp/file01.txt
[guest@yegor ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 окт  7 20:36 /tmp/file01.txt
[guest@yegor ~]$ chmod o+rw /tmp/file01.txt
[guest@yegor ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 окт  7 20:36 /tmp/file01.txt
[guest@yegor ~]$
```

Рис. 11: Создание файла /tmp/file01.txt

Исследование Sticky-бита

От пользователя guest2 попробуем прочитать файл /tmp/file01.txt. Далее попробуем дозаписать в файл /tmp/file01.txt слово test2, записать в файл /tmp/file01.txt слово test3, стерев при этом всю имеющуюся в файле информацию. После этого попробуем удалить данных файл

```
[guest@yegor ~]$ su guest2
Пароль:
[guest2@yegor guest]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@yegor guest]$ cat /tmp/file01.txt
test
[guest2@yegor guest]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@yegor guest]$ cat /tmp/file01.txt
test
[guest2@yegor guest]$ rm /tmp/file01.txt
rm: удалить защищённый от записи обычный файл '/tmp/file01.txt'? y
rm: невозможно удалить '/tmp/file01.txt': Операция не позволена
[guest2@yegor guest]$
```

Исследование Sticky-бита

От имени суперпользователя снимем атрибут t с директории /tmp. От пользователя guest2 проверим, что атрибута t у директории /tmp нет. Повторим предыдущие шаги. Теперь мы можем удалить файл

```
[root@yegor ~]# chmod -t /tmp
[root@yegor ~]# exit
выход
[guest2@yegor guest]$ ls -l / | grep tmp
drwxrwxrwx.  15 root root 4096 окт  7 20:48 tmp
[guest2@yegor guest]$ echo "test2" >> /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@yegor guest]$ echo "test2" > /tmp/file01.txt
bash: /tmp/file01.txt: Отказано в доступе
[guest2@yegor guest]$ ls -l / | grep tmp
drwxrwxrwx.  15 root root 4096 окт  7 20:48 tmp
[guest2@yegor guest]$ cat /tmp/file01.txt
test
[guest2@yegor guest]$ rm /tmp/file01.txt
rm: удалить защищённый от записи обычный файл '/tmp/file01.txt'? y
[guest2@yegor guest]$
```

Вывод

В ходе лабораторной работы мне удалось:

- Изучить механизмы изменения идентификаторов, применения SetUID- и Sticky-битов.
- Получить практических навыков работы в консоли с дополнительными атрибутами.
- Рассмотреть работу механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.