

Лабораторная работа №5

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

Галиева Аделина Руслановна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	15

Список иллюстраций

2.1	Подготовка	7
2.2	Программа simpleid.c	8
2.3	Результат программы simpleid	8
2.4	Программа simpleid2	9
2.5	Результат программы simpleid2	10
2.6	Программа readfile	11
2.7	Результат программы readfile	12
2.8	Результат программы readfile	13
2.9	Исследование Sticky-бит	14

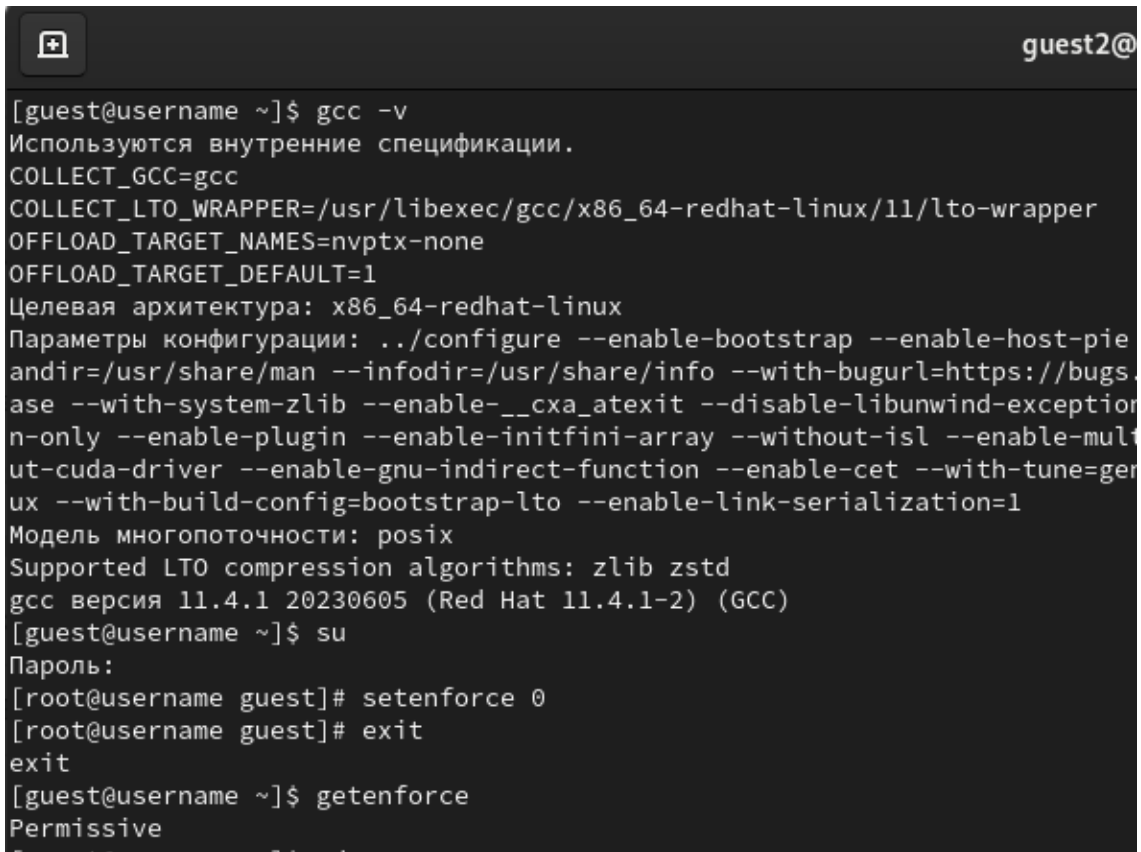
Список таблиц

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 лабораторной работы

1. Для выполнения части заданий требуются средства разработки приложений. Проверяем наличие установленного компилятора gcc -v: компилятор обнаружен.
2. Чтобы система защиты SELinux не мешала выполнению заданий работы, отключили систему запретов до очередной перезагрузки системы командой `setenforce 0`.
3. Команда `getenforce` вывела `Permissive`.



```
[guest@username ~]$ gcc -v
Используются внутренние спецификации.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/libexec/gcc/x86_64-redhat-linux/11/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none
OFFLOAD_TARGET_DEFAULT=1
Целевая архитектура: x86_64-redhat-linux
Параметры конфигурации: ../configure --enable-bootstrap --enable-host-pie
andir=/usr/share/man --infodir=/usr/share/info --with-bugurl=https://bugs.
ase --with-system-zlib --enable-__cxa_atexit --disable-libunwind-exception
n-only --enable-plugin --enable-initfini-array --without-isl --enable-mult
ut-cuda-driver --enable-gnu-indirect-function --enable-cet --with-tune=ger
ux --with-build-config=bootstrap-lto --enable-link-serialization=1
Модель многопоточности: posix
Supported LTO compression algorithms: zlib zstd
gcc версия 11.4.1 20230605 (Red Hat 11.4.1-2) (GCC)
[guest@username ~]$ su
Пароль:
[root@username guest]# setenforce 0
[root@username guest]# exit
exit
[guest@username ~]$ getenforce
Permissive
```

Рис. 2.1: Подготовка

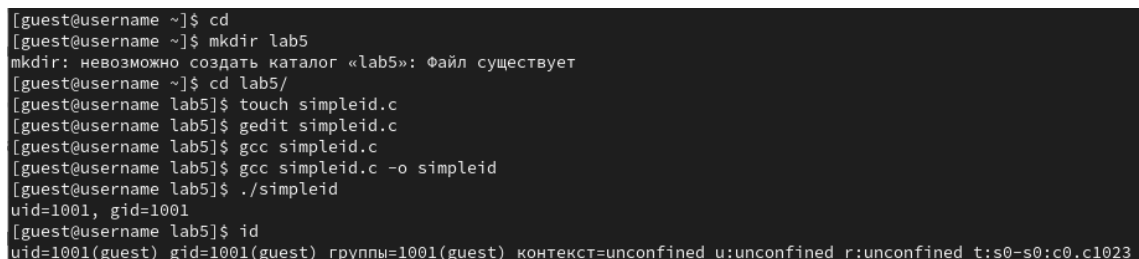
4. Входим в систему от имени пользователя guest.
5. Создаем команду simpleid.c.



```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 int main()
5 {
6     uid_t uid = geteuid();
7     gid_t gid = getegid();
8     printf("uid=%d, gid=%d\n", uid, gid);
9     return 0;
10 }
11
```

Рис. 2.2: Программа simpleid.c

6. Компилируем программу и убеждаемся, что файл программы создан.
7. Выполняем программу simpleid.
8. Выполняем системную программу id и gid совпадает в обеих программах.



```
[guest@username ~]$ cd
[guest@username ~]$ mkdir lab5
mkdir: невозможно создать каталог «lab5»: Файл существует
[guest@username ~]$ cd lab5/
[guest@username lab5]$ touch simpleid.c
[guest@username lab5]$ gedit simpleid.c
[guest@username lab5]$ gcc simpleid.c
[guest@username lab5]$ gcc simpleid.c -o simpleid
[guest@username lab5]$ ./simpleid
uid=1001, gid=1001
[guest@username lab5]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

Рис. 2.3: Результат программы simpleid

9. Усложняем программу, добавив вывод действительных идентификаторов.



```
1 #include <sys/types.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 int main()
5 {
6     uid_t e_uid = geteuid();
7     gid_t e_gid = getegid();
8     uid_t real_uid = getuid();
9     gid_t real_gid = getgid();
10    printf("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
11    printf("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
12    return 0;
13 }
```

Рис. 2.4: Программа simpleid2

10. Компилируем и запускаем simpleid2.c.
11. От имени суперпользователя выполняем команды: `chown root:guest /home/guest/simpleid2`, `chmod u+s /home/guest/simpleid2`
12. Используем `su` для повышения прав до суперпользователя.
13. Выполняем проверку правильности установки новых атрибутов и смены владельца файла simpleid2.
14. Запускаем simpleid2 и `id`. Результат выполнения программ теперь немного отличается.
15. Проделаем тоже самое относительно SetGID-бита.

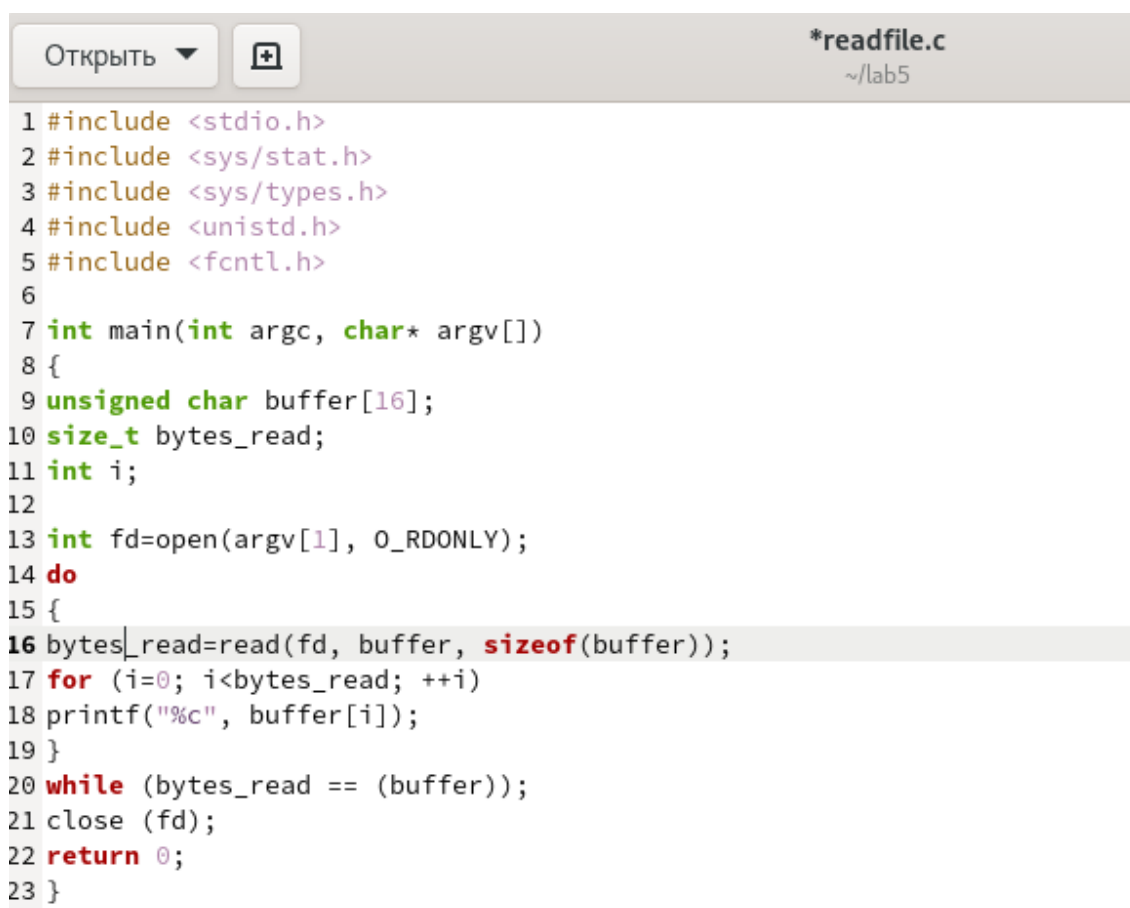
```

[guest@username lab5]$ touch simpleid2.c
[guest@username lab5]$ gedit simpleid2.c
[guest@username lab5]$ gcc simpleid2.c
[guest@username lab5]$ gcc simpleid2.c -o simpleid2
[guest@username lab5]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@username lab5]$ su
Пароль:
[root@username lab5]# chown root: guest simpleid2
chown: невозможно получить доступ к 'guest': Нет такого файла или каталога
[root@username lab5]# chmod u+s simpleid2
[root@username lab5]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@username lab5]# id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[root@username lab5]# chmod g+s simpleid2
[root@username lab5]# ./simpleid2
e_uid=0, e_gid=0
real_uid=0, real_gid=0
[root@username lab5]# exit
exit
[guest@username lab5]$ ./simpleid2
e_uid=0, e_gid=0
real_uid=1001, real_gid=1001

```

Рис. 2.5: Результат программы simpleid2

16. Создаем программу readfile.c.



```
1 #include <stdio.h>
2 #include <sys/stat.h>
3 #include <sys/types.h>
4 #include <unistd.h>
5 #include <fcntl.h>
6
7 int main(int argc, char* argv[])
8 {
9     unsigned char buffer[16];
10    size_t bytes_read;
11    int i;
12
13    int fd=open(argv[1], O_RDONLY);
14    do
15    {
16        bytes_read=read(fd, buffer, sizeof(buffer));
17        for (i=0; i<bytes_read; ++i)
18            printf("%c", buffer[i]);
19    }
20    while (bytes_read == (buffer));
21    close (fd);
22    return 0;
23 }
```

Рис. 2.6: Программа readfile

17. Откомпилируем её.
18. Сменили владельца у файла readfile.c и меняем права так, чтобы только суперпользователь (root) мог прочитать его, а guest не мог.
19. Проверяем, что пользователь guest не может прочитать файл readfile.c.
20. Меняем у программы readfile владельца и установите SetU'D-бит.
21. Проверяем, может ли программа readfile прочитать файл readfile.c.
22. Проверяем, может ли программа readfile прочитать файл /etc/shadow.

```

[guest@username lab5]$ touch readfile.c
[guest@username lab5]$ gedit readfile.c
[guest@username lab5]$ gedit readfile.c
[guest@username lab5]$ su
Пароль:
[root@username lab5]# gcc readfile.c
readfile.c: В функции «main»:
readfile.c:20:19: предупреждение: сравнение указателя и целого
   20 | while (bytes_read == (buffer));
      |                   ^~
[root@username lab5]# gcc readfile.c -o readfile
readfile.c: В функции «main»:
readfile.c:20:19: предупреждение: сравнение указателя и целого
   20 | while (bytes_read == (buffer));
      |                   ^~
[root@username lab5]# su
[root@username lab5]# chown root:root readfile
[root@username lab5]# chmod u+s readfile
[root@username lab5]# exit
exit
[root@username lab5]# su
[root@username lab5]# chmod -rwx readfile.c
[root@username 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
    {

```

Рис. 2.7: Результат программы readfile

31. От пользователя guest2 пробуем удалить файл /tmp/file01.txt командой rm /tmp/file01.txt. Получаем отказ.
32. От суперпользователя проверили, что атрибута t у директории /tmp нет.
33. Повторяем предыдущие шаги. Получилось удалить файл.
34. Удалось удалить файл от имени пользователя, не являющегося его владельцем.
35. Повышаем свои права до суперпользователя и возвращаем атрибут t на директорию /tmp.

```
root:$6$.Cqhmypl[root@username lab5]# cd /tmp
[root@username tmp]# echo test >> file01.txt
[root@username tmp]# chmod g+rwX file01.txt
[root@username tmp]# su guest2
[guest2@username tmp]$ echo test2 >> file01.txt
bash: file01.txt: Отказано в доступе
[guest2@username tmp]$ cat file01.txt
test
[guest2@username tmp]$ echo 123 > file01.txt
bash: file01.txt: Отказано в доступе
[guest2@username tmp]$ rm file01.txt
rm: удалить защищённый от записи обычный файл 'file01.txt'?
[guest2@username tmp]$ su
Пароль:
[root@username tmp]# chmod -t /tmp
[root@username tmp]# exit
exit
[guest2@username tmp]$ rm file01.txt
rm: удалить защищённый от записи обычный файл 'file01.txt'?
[guest2@username tmp]$
```

Рис. 2.9: Исследование Sticky-бит

3 Выводы

В ходе выполнения лабораторной работы я изучила механизмы изменения идентификаторов, применения SetUID- и Sticky-битов. Получила практические навыки работы в консоли с дополнительными атрибутами. Рассмотрела работы механизма смены идентификатора процессов пользователей, а также влияние бита Sticky на запись и удаление файлов.