Презентация по лабораторной работе N4

Основы информационной безопасноти

Соловьев Богдан НКАбд-04-23

Содержание

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

Теоретическая часть

Помимо прав администратора для выполнения части заданий потребуются средства разработки приложений. В частности, при подготовке стенда
следует убедиться, что в системе установлен компилятор gcc (для этого, например, можно ввести команду gcc -v). Если же gcc не установлен, то его
необходимо установить, например, командой
yum install gcc

которая определит зависимости и установит следующие пакеты:
gcc, cloogppl, cpp, glibc-devel, glibc-headers, kernel-headers,
libgomp, ppl, cloog-ppl,
cpp, gcc, glibc-devel, glibc-headers, kernel-headers, libgomp,
libstdc++-devel,
mpfr, ppl, glibc, glibc-common, libgcc, libstdc++.

Файловая система, где располагаются домашние директории и файлы
пользователей (в частности, пользователя guest), не должна быть смонтирована с опцией nosuid.
Так как программы с установленным битом SetUID могут представлять
большую брешь в системе безопасности, в современных системах используются дополнительные механизмы защиты.

Проследите, чтобы система
защиты SELinux не мешала выполнению заданий работы. Если вы не знаете, что это такое, просто отключите систему запретов до очередной перезагрузки системы командой
setenforce 0
После этого команда getenforce должна выводить Permissive. В этой
работе система SELinux рассматриваться не будет.

Такое решение подойдёт лишь для простых случаев. Если говорить про
пример выше, то компилирование одного файла из двух шагов можно сократить вообще до одного, например:
gcc file.c
В этом случае готовая программа будет иметь называние a.out.
Механизм компилирования программ в данной работе не мог быть не
рассмотрен потому, что использование программ, написанных на bash, для
изучения SetUID- и SetGID- битов, не представляется возможным.

Связано это с тем, что любая bash-программа интерпретируется в процессе своего
выполнения, т.е. существует сторонняя программа-интерпретатор, которая
выполняет считывание файла сценария и выполняет его последовательно.

Сам интерпретатор выполняется с правами пользователя, его запустившего,
а значит, и выполняемая программа использует эти права.
При этом интерпретатору абсолютно всё равно, установлены SetUID-,
SetGID-биты у текстового файла сценария, атрибут разрешения запуска «x»
или нет. Важно, чтобы был установлен лишь атрибут, разрешающий чтение
«r».

Также не важно, был ли вызван интерпретатор из командной строки
(запуск файла, как bash file1.sh), либо внутри файла была указана строчка
#!/bin/bash.

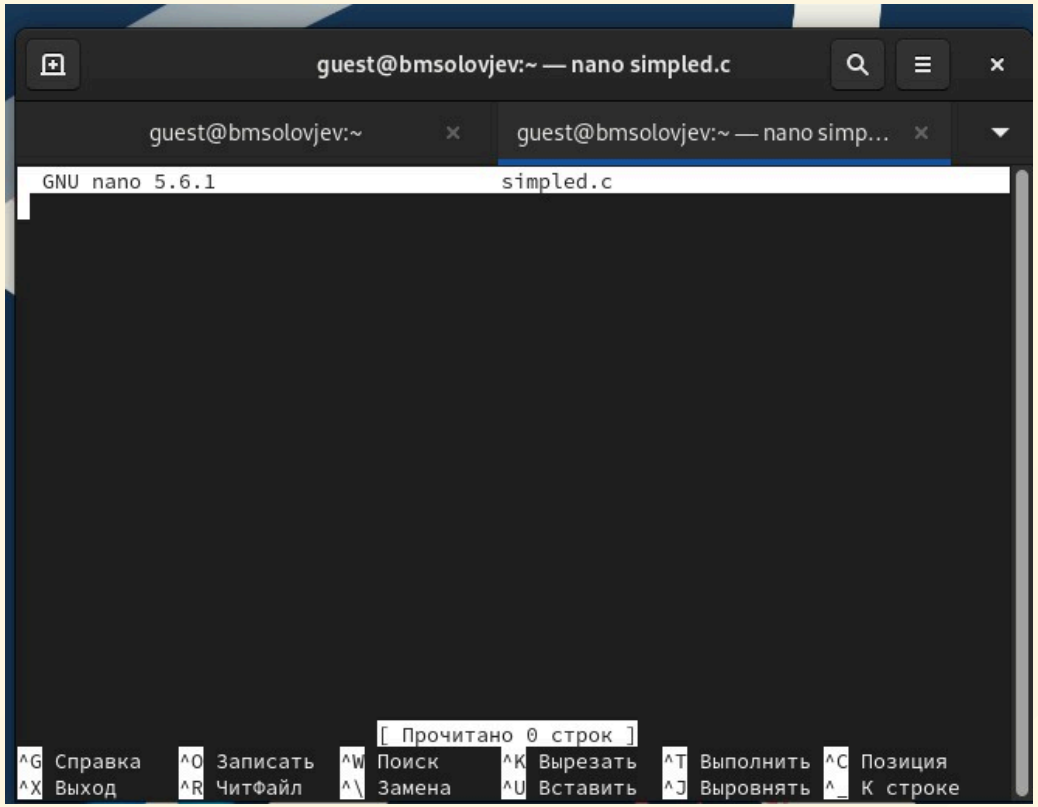
Логично спросить: если установление SetUID- и SetGID- битов на сценарий не приводит к нужному результату как с исполняемыми файлами,
то что мешает установить эти биты на сам интерпретатор?

Ничего не мешает, только их установление приведёт к тому, что, так как владельцем
/bin/bash является root:
ls -l /bin/bash
все сценарии, выполняемые с использованием /bin/bash, будут иметь возможности суперпользователя — совсем не тот результат, который хотелось
бы видеть.

Выполнение лабораторной работы

Осуществляю вход от имени пользователя guest

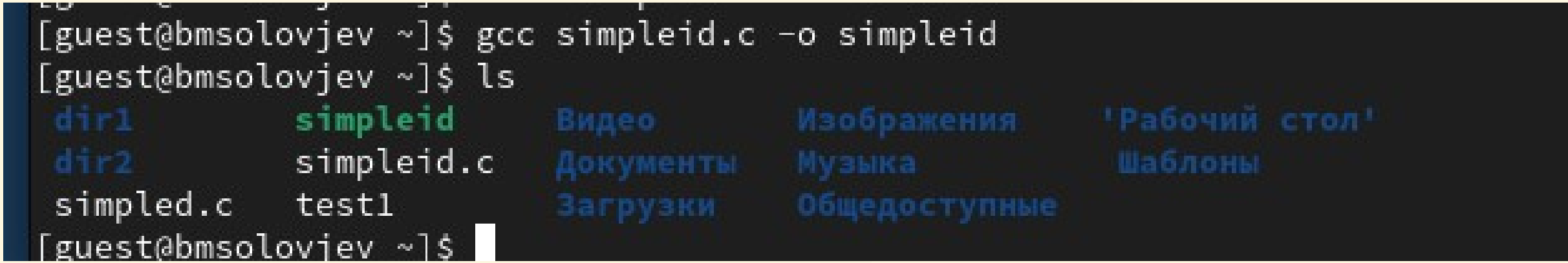
```
[guest@bmsolovjev ~]$ su guest
Пароль:
[guest@bmsolovjev ~]$
```



Создаю файл simpleid и начинаю его редактировать

```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;
}
```

вписываю в созданный файл этот код



Компилирую файл и проверяю, что исполняемый файл создался

```
[guest@bmsolovjev ~]$ ./simpleid
uid=1001, gid=1001
[guest@bmsolovjev ~]$ id
uid=1001(guest) gid=1001(guest) группы=1001(guest),10(wh
_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[guest@bmsolovjev ~]$
```

Запускаю исполняемый файл, от команды id вывод файла отличается только тем, что он выводит меньше информации

```
[guest@bmsolovjev ~]$ nano simpleid2.c
[guest@bmsolovjev ~]$ gcc simpleid2.c -o simpleid2
[guest@bmsolovjev ~]$ ./simpled2
bash: ./simpled2: Нет такого файла или каталога
[guest@bmsolovjev ~]$ ./simpleid2
e_uid=1001, e_gid=1001
real_uid=1001, real_gid=1001
[guest@bmsolovjev ~]$
```

Теперь создаю файл simpleid2

simpleid2 содержит следующий код:

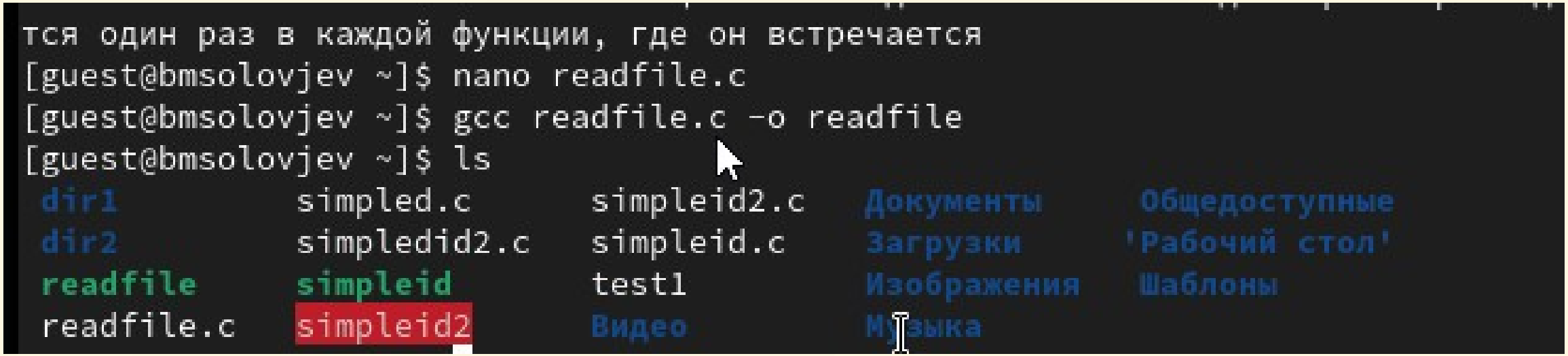
```c
#include <sys/types.h>
#include <unistd.h>
#include <stdio.h>
int
main ()
{
uid_t real_uid = getuid ();
uid_t e_uid = geteuid ();
gid_t real_gid = getgid ();
gid_t e_gid = getegid () ;
printf ("e_uid=%d, e_gid=%d\n", e_uid, e_gid);
printf ("real_uid=%d, real_gid=%d\n", real_uid, real_gid);
return 0;
}
```

с помощью chown изменяю владельца на супеопользователя и меняю права доступа с помощью chmod

```
[guest@bmsolovjev ~]$ sudo chown root:guest home/guest/simpleid2
[sudo] пароль для guest:
chown: невозможно получить доступ к 'home/guest/simpleid2': Нет так
 каталога
[guest@bmsolovjev ~]$ sudo chmod u+s /home/guest/simpleid2
[guest@bmsolovjev ~]$ sudo ls -l /home/guest/simpleid2
-rwsr-xr-x. 1 guest guest 17656 апр  4 16:15 /home/guest/simpleid2
[guest@bmsolovjev ~]$
```

```
[guest@bmsolovjev ~]$ sudo /home/guest/simpleid2
e_uid=1001, e_gid=0
real_uid=0, real_gid=0
[guest@bmsolovjev ~]$ sudp id
bash: sudp: команда не найдена...
[guest@bmsolovjev ~]$ sudo id
uid=0(root) gid=0(root) группы=0(root) контекст=unconfined_u:unconfined_r:unconf
ined_t:s0-s0:c0.c1023
[guest@bmsolovjev ~]$
```

Полученная программа вывела всё равно не всю информацию



Создаю и компилирую файл readfile.c

```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;
}
```

От имени суперпользователя меняю владельца файла readfile.c и изменяю права доступа для guest так, чтобы можно было прочесть содержимое файла, но получаем отказ в доступе

```
[guest@bmsolovjev ~]$ sudo chown root:guest /home/guest/readfile.c
[guest@bmsolovjev ~]$ sudo chmod u+s /home/guest/readfile.c
[guest@bmsolovjev ~]$ sudo chmod 700 /home/guest/readfile.c
[guest@bmsolovjev ~]$ sudo chmod -r /home/guest/readfile.c
[guest@bmsolovjev ~]$ cat readfile.c
cat: readfile.c: Отказано в доступе
[guest@bmsolovjev ~]$ ./readfile readfile.c
@@
```

```
Ошибка сегментирования (стек памяти сброшен на диск)
[guest@bmsolovjev ~]$ sudo /home/guest/readfile /etc/shadow
root:$6$M3j6SGTt1xm8IShN$redjdieMU5a1X6HjC6xD3BPzjR.7MsGdqozjyO9kaYpIHSQIa94lC5Z
AFJBjPpvqmSi/xpmylOOGI2dSi3TTe0::0:99999:7:::
bin:*:19820:0:99999:7:::
daemon:*:19820:0:99999:7:::
adm:*:19820:0:99999:7:::
lp:*:19820:0:99999:7:::
sync:*:19820:0:99999:7:::
shutdown:*:19820:0:99999:7:::
halt:*:19820:0:99999:7:::
mail:*:19820:0:99999:7:::
operator:*:19820:0:99999:7:::
```

От имени суперпользователя всё читается

```
[guest@bmsolovjev ~]$ ls -l / | grep tmp
drwxrwxrwt.  16 root root 4096 апр  4 20:45 tmp
[guest@bmsolovjev ~]$ echo "test" > /tmp/file01.txt
[guest@bmsolovjev ~]$ ls -l /tmp/file01.txt
-rw-r--r--. 1 guest guest 5 апр  4 20:47 /tmp/file01.txt
[guest@bmsolovjev ~]$ chmod o+rw /tmp/file01.txt
[guest@bmsolovjev ~]$ ls -l /tmp/file01.txt
-rw-r--rw-. 1 guest guest 5 апр  4 20:47 /tmp/file01.txt
[guest@bmsolovjev ~]$
```

Проверяем папку tmp на налицие атрибута Sticky (если есть буква t в выводе, то атрмбут присутсвует). t есть в выоде, потом создаю файл с текстом и разрешаю его читать другим пользователям

Разрешено чтение, запись

Выводы

Я изучил механизм изменения идентификаторов