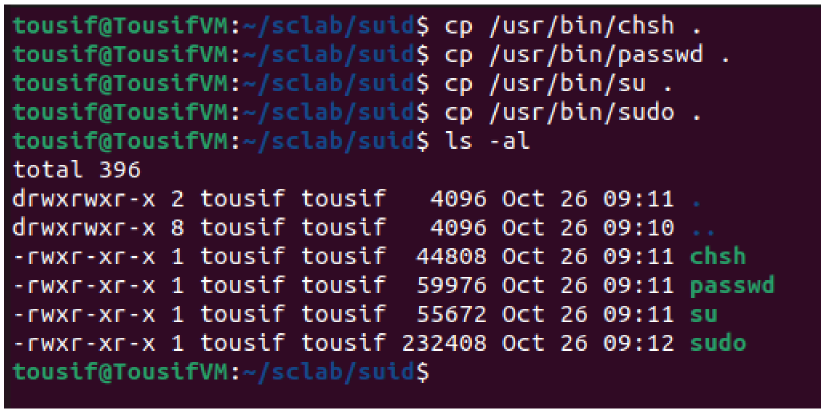
Name : Mohammed Tousif
Roll : CB.EN.P2CYS22008

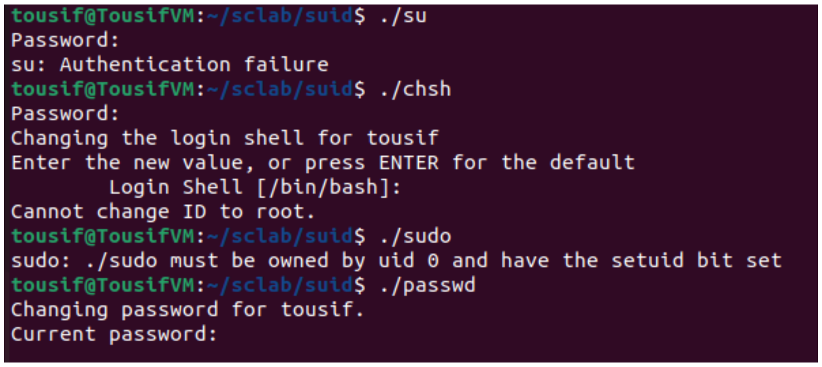
SECURE CODING LAB
Set-UID

1. Figure out why"passwd","chsh","su", and"sudo"commands need to be Set-UID programs. What will happen if they are not? If you are not familiar with these programs, you should firstlearn what they can do by reading their manuals. Please copy these commands to your own directory;the copies will not be Set-UIDprograms. Run the copied programs, and observe what happens.



```
tousif@TousifVM:~/sclab/suid$ cp /usr/bin/chsh .
tousif@TousifVM:~/sclab/suid$ cp /usr/bin/passwd .
tousif@TousifVM:~/sclab/suid$ cp /usr/bin/su .
tousif@TousifVM:~/sclab/suid$ cp /usr/bin/sudo .
tousif@TousifVM:~/sclab/suid$ ls -al
total 396
drwxrwxr-x 2 tousif tousif   4096 Oct 26 09:11 .
drwxrwxr-x 8 tousif tousif   4096 Oct 26 09:10 ..
-rwxr-xr-x 1 tousif tousif  44808 Oct 26 09:11 chsh
-rwxr-xr-x 1 tousif tousif  59976 Oct 26 09:11 passwd
-rwxr-xr-x 1 tousif tousif  55672 Oct 26 09:11 su
-rwxr-xr-x 1 tousif tousif 232408 Oct 26 09:12 sudo
tousif@TousifVM:~/sclab/suid$
```

All the copied programs are not Set-UID programs. Thus , the s bits are missing and it can’t run due to lack of root privilege. As per the security measures, it prevents the root privilege.

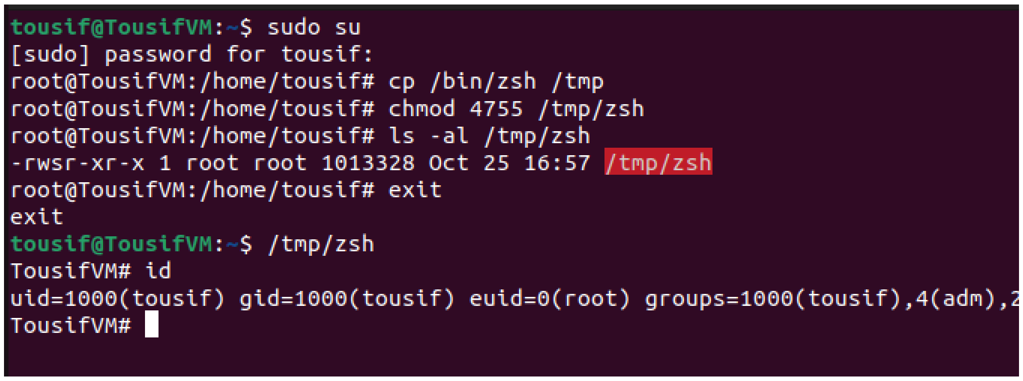


```
tousif@TousifVM:~/sclab/suid$ ./su
Password:
su: Authentication failure
tousif@TousifVM:~/sclab/suid$ ./chsh
Password:
Changing the login shell for tousif
Enter the new value, or press ENTER for the default
        Login Shell [/bin/bash]:
Cannot change ID to root.
tousif@TousifVM:~/sclab/suid$ ./sudo
sudo: ./sudo must be owned by uid 0 and have the setuid bit set
tousif@TousifVM:~/sclab/suid$ ./passwd
Changing password for tousif.
Current password:
```

As shown in above image , we can't run those programs as it needs root privilege to run.

2. Run Set-UID shell programs in Linux, and describe and explain your observations.

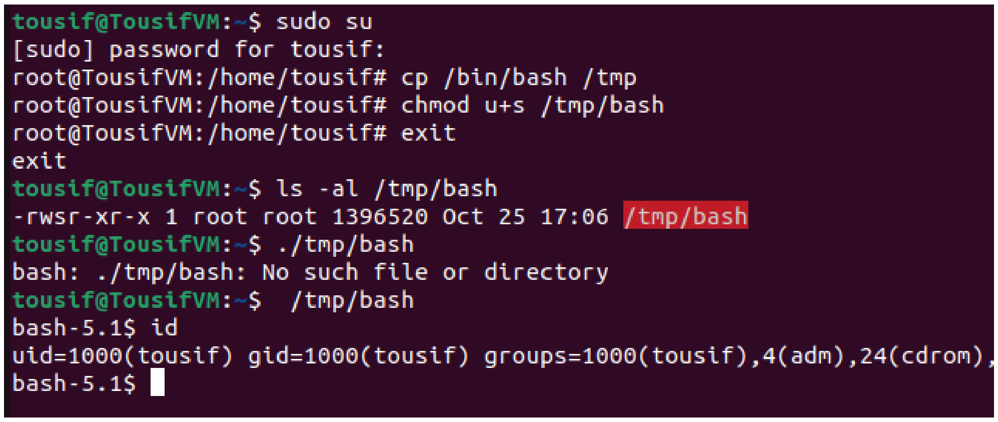
(a) Login as root, copy /bin/zsh to /tmp, and make it a set-root-uid program with permission 4755. Then login as a normal user, and run /tmp/zsh. Will you get root privilege?



```
tousif@TousifVM:~$ sudo su
[sudo] password for tousif:
root@TousifVM:/home/tousif# cp /bin/zsh /tmp
root@TousifVM:/home/tousif# chmod 4755 /tmp/zsh
root@TousifVM:/home/tousif# ls -al /tmp/zsh
-rwsr-xr-x 1 root root 1013328 Oct 25 16:57 /tmp/zsh
root@TousifVM:/home/tousif# exit
exit
tousif@TousifVM:~$ /tmp/zsh
TousifVM# id
uid=1000(tousif) gid=1000(tousif) euid=0(root) groups=1000(tousif),4(adm),2
TousifVM#
```

As euid=0(root) , it denotes that we got root privilege.

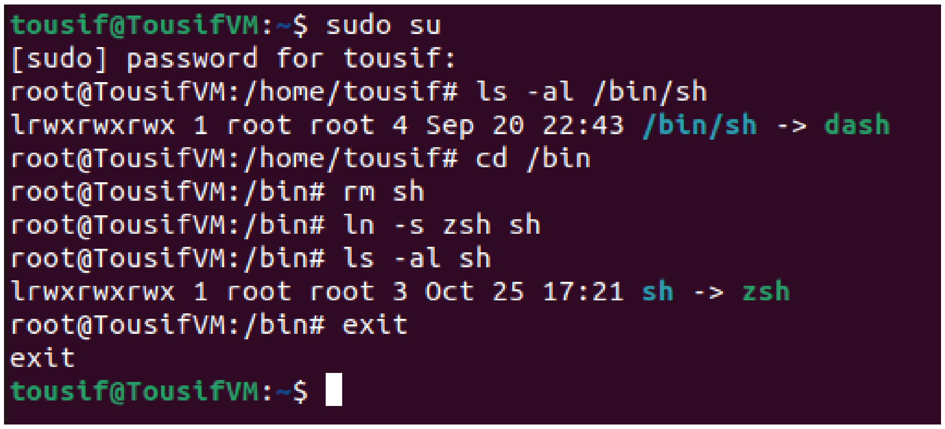
(b) Instead of copying /bin/zsh, this time, copy /bin/bash to /tmp, make it a set-root-uid program. Run /tmp/bash as a normal user. will you get root privilege? Please describe and explain your observation.



```
tousif@TousifVM:~$ sudo su
[sudo] password for tousif:
root@TousifVM:/home/tousif# cp /bin/bash /tmp
root@TousifVM:/home/tousif# chmod u+s /tmp/bash
root@TousifVM:/home/tousif# exit
exit
tousif@TousifVM:~$ ls -al /tmp/bash
-rwsr-xr-x 1 root root 1396520 Oct 25 17:06 /tmp/bash
tousif@TousifVM:~$ ./tmp/bash
bash: ./tmp/bash: No such file or directory
tousif@TousifVM:~$  /tmp/bash
bash-5.1$ id
uid=1000(tousif) gid=1000(tousif) groups=1000(tousif),4(adm),24(cdrom),
bash-5.1$
```

But for /tmp/bash file, while running it as normal user. we didn't get the root privilege due to some security measures.

3. As you can find out from the previous task, /bin/bash has certain built-in protection that prevent the abuse of the Set-UID mechanism. To see the life before such a protection scheme was implemented, we are going to use a different shell program called /bin/zsh. In some Linux distributions (such as Fedora and Ubuntu),/bin/shis actually a symbolic link to /bin/bash. To use zsh, we need to link /bin/sh to /bin/zsh. The following instructions describe how to change the default shell to zsh.



```
tousif@TousifVM:~$ sudo su
[sudo] password for tousif:
root@TousifVM:/home/tousif# ls -al /bin/sh
lrwxrwxrwx 1 root root 4 Sep 20 22:43 /bin/sh -> dash
root@TousifVM:/home/tousif# cd /bin
root@TousifVM:/bin# rm sh
root@TousifVM:/bin# ln -s zsh sh
root@TousifVM:/bin# ls -al sh
lrwxrwxrwx 1 root root 3 Oct 25 17:21 sh -> zsh
root@TousifVM:/bin# exit
exit
tousif@TousifVM:~$ 
```

As stated, first we remove the sh binary from /bin. Create a link by ln command. sh - > zsh denotes that link is created for sh binary.

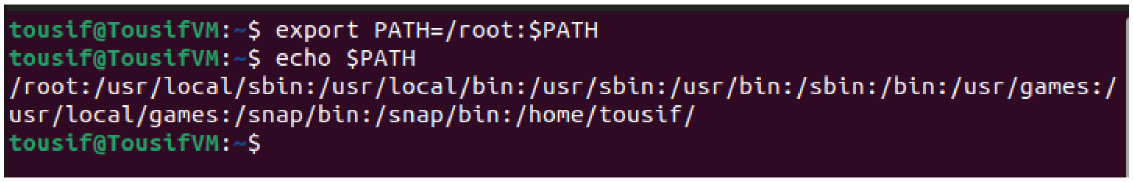
4.The PATH environment variable
The system(const char*cmd) library function can be used to execute a command within a program. The way system(cmd) works is to invoke the /bin/sh program, and then let the shell program to execute cmd. Because of the shell program invoked, calling system()within a Set-UID program is extremely dangerous. This is because the actual behavior of the shell program can be affected by environment variables, such as PATH; these environment variables are under user's control. By changing these variables, malicious users can control the behavior of the Set-UID program. In bash, you can change the PATH environment variable in the following way.

The Set-UID program below is supposed to execute the /bin/ls command; however, the programmer only uses the relative path for the ls command, rather than the absolute path:

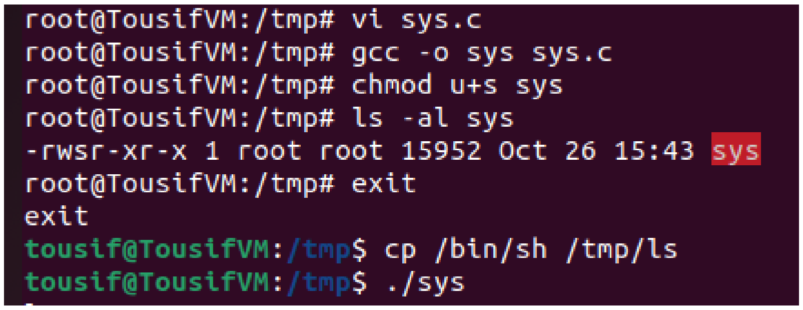
```
#include<stdlib.h>
int main()
{
system("ls");
return 0;
}
```

(a) Can you let this Set-UID program (owned by root) run your code instead of /bin/ls? If you can, is your code running with the root privilege?

The above program is a Set-UID program which will be run as root. Thus , copying /bin/sh as ls and change the value of PATH variable through root directory path, then we can run our ls binary which is in /bin/sh binary.



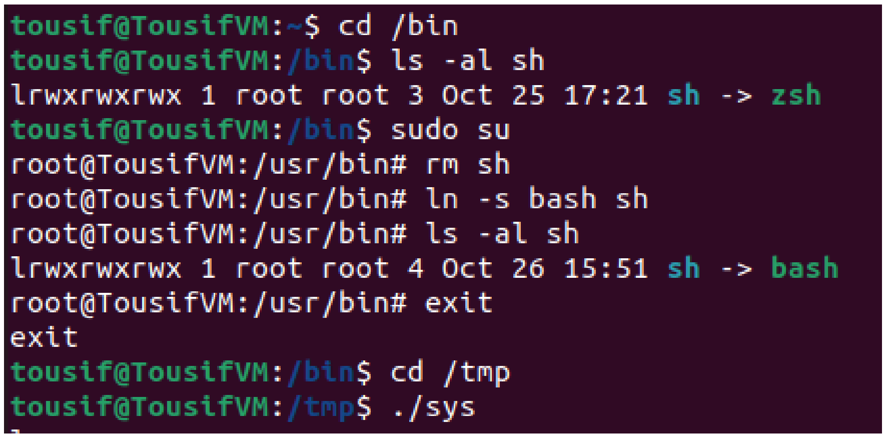
```
tousif@TousifVM:~$ export PATH=/root:$PATH
tousif@TousifVM:~$ echo $PATH
/root:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/
usr/local/games:/snap/bin:/snap/bin:/home/tousif/
tousif@TousifVM:~$
```



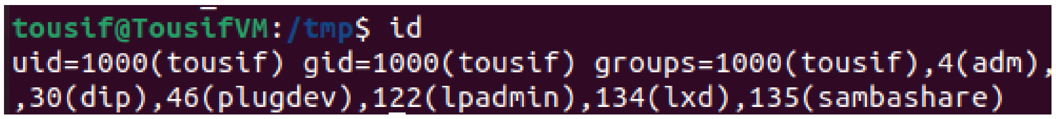
```
root@TousifVM:/tmp# vi sys.c
root@TousifVM:/tmp# gcc -o sys sys.c
root@TousifVM:/tmp# chmod u+s sys
root@TousifVM:/tmp# ls -al sys
-rwsr-xr-x 1 root root 15952 Oct 26 15:43 sys
root@TousifVM:/tmp# exit
exit
tousif@TousifVM:/tmp$ cp /bin/sh /tmp/ls
tousif@TousifVM:/tmp$ ./sys
```

The above program has root privilege as it is running in root shell and owned by root.

(b) Now, change /bin/sh so it points back to /bin/bash, and repeat the above attack. Can you still get the root privilege? Describe and explain your observations.


```
tousif@TousifVM:~$ cd /bin
tousif@TousifVM:/bin$ ls -al sh
lrwxrwxrwx 1 root root 3 Oct 25 17:21 sh -> zsh
tousif@TousifVM:/bin$ sudo su
root@TousifVM:/usr/bin# rm sh
root@TousifVM:/usr/bin# ln -s bash sh
root@TousifVM:/usr/bin# ls -al sh
lrwxrwxrwx 1 root root 4 Oct 26 15:51 sh -> bash
root@TousifVM:/usr/bin# exit
exit
tousif@TousifVM:/bin$ cd /tmp
tousif@TousifVM:/tmp$ ./sys
```

Now we changed it back to bash and run the program. This time we didn't have root privilege.

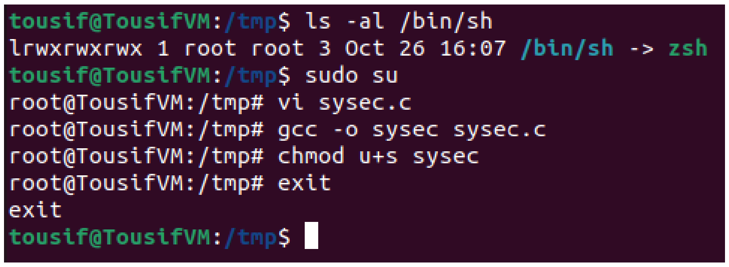

```
tousif@TousifVM:/tmp$ id
uid=1000(tousif) gid=1000(tousif) groups=1000(tousif),4(adm),
,30(dip),46(plugdev),122(lpadmin),134(lxd),135(sambashare)
```

5. The difference between system() and execve(). Before you work on this task, please make sure that /bin/sh is pointed to /bin/zsh.
Background:Bob works for an auditing agency, and he needs to investigate a company for a suspected fraud. For the investigation purpose, Bob needs to be able to read all the files in the company's Unixsystem; on the other hand, to protect the integrity of the system, Bob should not be able to modify any file. To achieve this goal, Vince, the superuser of the system, wrote a special set-root-uidprogram (see below), and then gave the executable permission to Bob. This program requires Bob to type a file name at the command line, and then it will run /bin/cat to display the specified file.

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char*argv[])
{
char*v[3];
if(argc < 2)
{
printf("Please type a file name.\n");
return 1;
}
v[0] = "/bin/cat";
v[1] = argv[1];
v[2] = 0;
/*Set q = 0 for Question a, and q = 1 for Question b*/
int q = 0;
if (q == 0)
{
char*command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
sprintf(command, "%s %s", v[0], v[1]);
system(command);
}
else
execve(v[0], v, 0);
return 0 ;
}
```

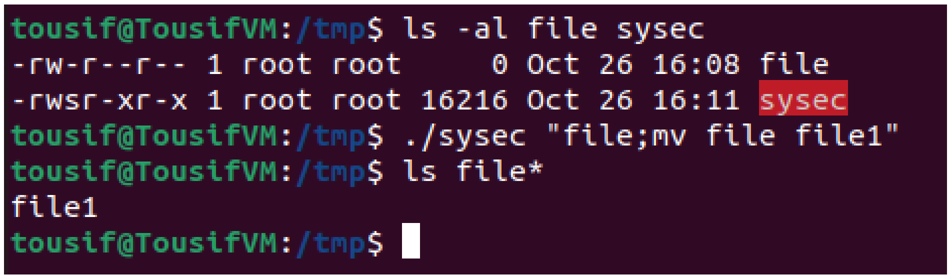
Since the program is running as a root, it can display any file Bob specifies. However, since the pro-gram has no write operations, Vince is very sure that Bob cannot use this special program to modifyany file.

(a) Setq= 0in the program. This way, the program will use system()to invoke the command.Is this program safe? If you were Bob, can you compromise the integrity of the system?
For example, can you remove any file that is not writable to you? (Hint: remember that system()actually invokes /bin/sh, and then runs the command within the shell environment. We have tried the environment variable in the previous task; here let us try a different attack. Please pay attention to the special characters used in a normal shell environment).



```
tousif@TousifVM:/tmp$ ls -al /bin/sh
lrwxrwxrwx 1 root root 3 Oct 26 16:07 /bin/sh -> zsh
tousif@TousifVM:/tmp$ sudo su
root@TousifVM:/tmp# vi sysec.c
root@TousifVM:/tmp# gcc -o sysec sysec.c
root@TousifVM:/tmp# chmod u+s sysec
root@TousifVM:/tmp# exit
exit
tousif@TousifVM:/tmp$
```

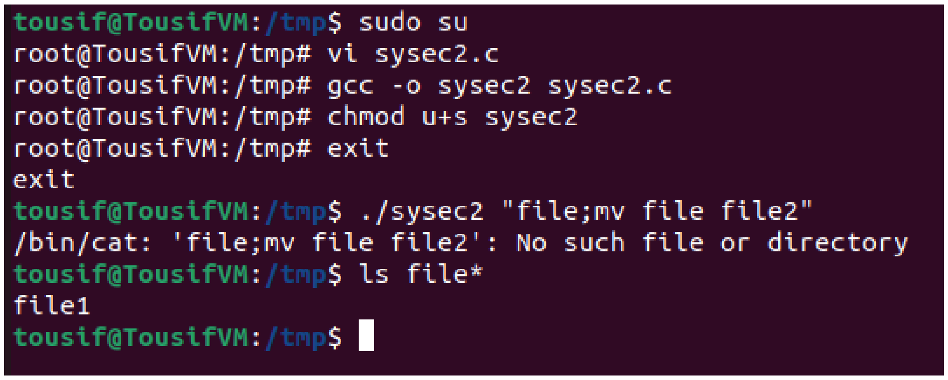
Setting q=0 in the program. the program is ready to run with set-UID bit. So that system() is invoked.



```
tousif@TousifVM:/tmp$ ls -al file sysec
-rw-r--r-- 1 root root     0 Oct 26 16:08 file
-rwsr-xr-x 1 root root 16216 Oct 26 16:11 sysec
tousif@TousifVM:/tmp$ ./sysec "file;mv file file1"
tousif@TousifVM:/tmp$ ls file*
file1
tousif@TousifVM:/tmp$
```

As you can see in above image, The "sysec" file is not safe. i.e. Bob can read , write and move files which only root user can.

(b) Set q= 1 in the program. This way, the program will use execve() to invoke the command. Do your attacks in task (a) still work? Please describe and explain your observations.

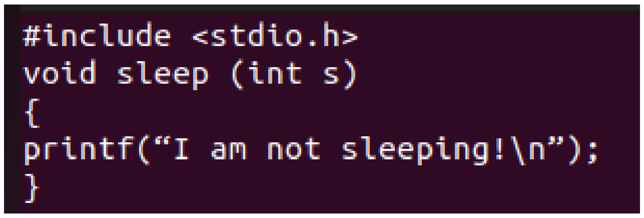


```
tousif@TousifVM:/tmp$ sudo su
root@TousifVM:/tmp# vi sysec2.c
root@TousifVM:/tmp# gcc -o sysec2 sysec2.c
root@TousifVM:/tmp# chmod u+s sysec2
root@TousifVM:/tmp# exit
exit
tousif@TousifVM:/tmp$ ./sysec2 "file;mv file file2"
/bin/cat: 'file;mv file file2': No such file or directory
tousif@TousifVM:/tmp$ ls file*
file1
tousif@TousifVM:/tmp$
```

Here we set q=1 in the program. It calls execve() instead of system(). Thus execve() will take that “file;mv file file2” and assume it as a folder name and returns as : No such file or directory.

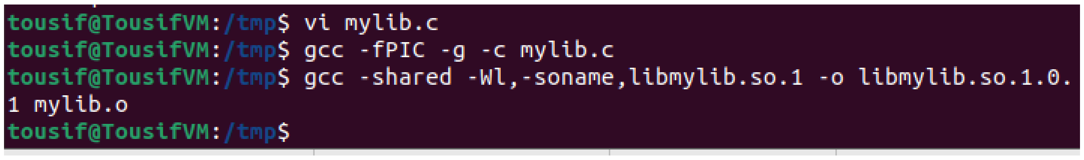
6.The LDPRELOAD environment variable.
To make sure Set-UID programs are safe from the manipulation of the LD_PRELOAD environment variable, the runtime linker (ld.so) will ignore this environment variable if the program is a Set-UID root program, except for some conditions. We will figure out what these conditions are in this task.

(a) Let us build a dynamic link library. Create the following program, and name it mylib.c. It basically overrides the sleep() function in libc:



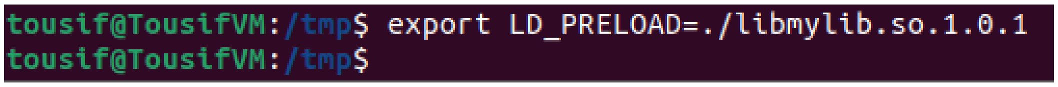
```
#include <stdio.h>
void sleep (int s)
{
printf(“I am not sleeping!\n”);
}
```

(b) We can compile the above program using the following commands (in the-Wl argument, the third character is l , not one; in the -lc argment, the second character is l):



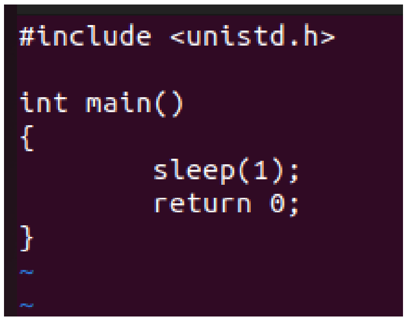
```
tousif@TousifVM:/tmp$ vi mylib.c
tousif@TousifVM:/tmp$ gcc -fPIC -g -c mylib.c
tousif@TousifVM:/tmp$ gcc -shared -Wl,-soname,libmylib.so.1 -o libmylib.so.1.0.
1 mylib.o
tousif@TousifVM:/tmp$
```

(c) Now, set the LD_PRELOAD environment variable:

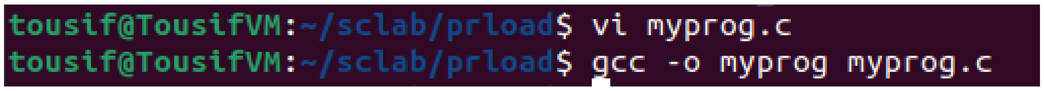


```
tousif@TousifVM:/tmp$ export LD_PRELOAD=./libmylib.so.1.0.1
tousif@TousifVM:/tmp$
```

d) Finally, compile the following program myprog (put this program in the same directory as libmylib.so.1.0.1):

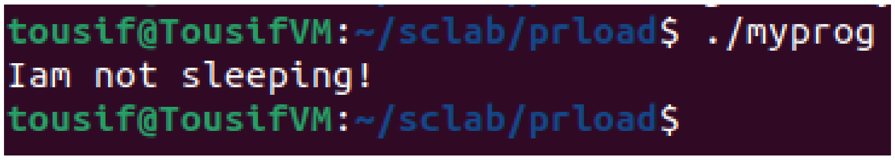

```
#include <unistd.h>

int main()
{
        sleep(1);
        return 0;
}
~
~
```

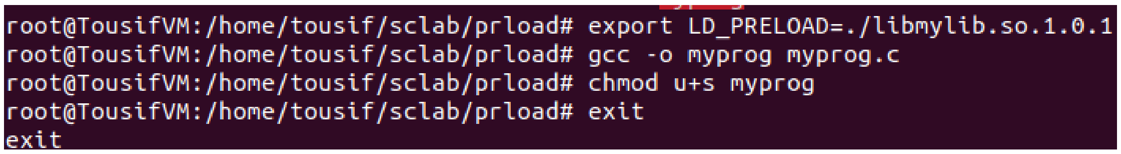

```
tousif@TousifVM:~/sclab/prload$ vi myprog.c
tousif@TousifVM:~/sclab/prload$ gcc -o myprog myprog.c
```

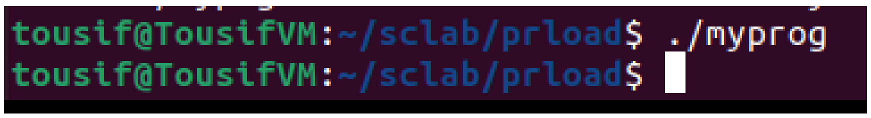
Please run myprog under the following conditions, and observe what happens. Based on your ob-servations, tell us when the runtime linker will ignore the LDPRELOAD environment variable, and explain why.

• Make myprog a regular program, and run it as a normal user.


```
tousif@TousifVM:~/sclab/prload$ ./myprog
Iam not sleeping!
tousif@TousifVM:~/sclab/prload$
```

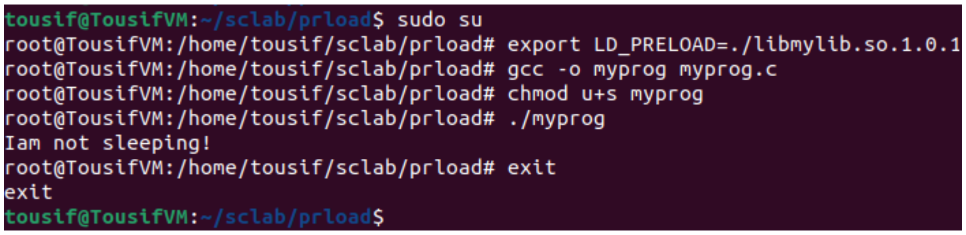
• **Make myprog a Set-UID root program, and run it as a normal user**.


```
root@TousifVM:/home/tousif/sclab/prload# export LD_PRELOAD=./libmylib.so.1.0.1
root@TousifVM:/home/tousif/sclab/prload# gcc -o myprog myprog.c
root@TousifVM:/home/tousif/sclab/prload# chmod u+s myprog
root@TousifVM:/home/tousif/sclab/prload# exit
exit
```


```
tousif@TousifVM:~/sclab/prload$ ./myprog
tousif@TousifVM:~/sclab/prload$
```

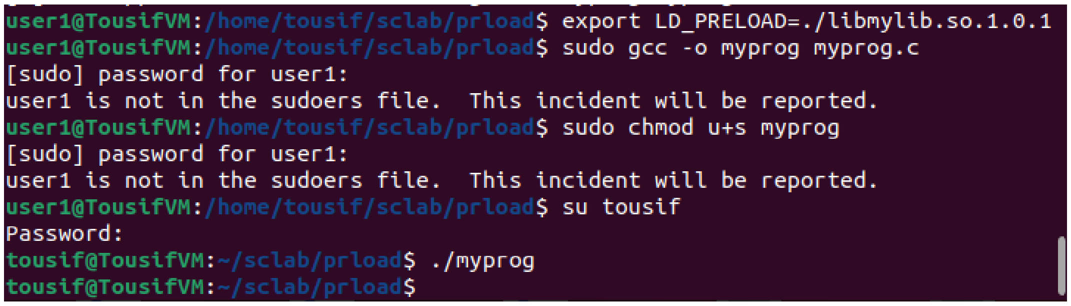
In this case , it will ignore the LD_PRELOAD variable and takes sleep function as default sleep(). Thus it executes sleep().

•Make myprog a Set-UID root program, and run it in the root account.



```
tousif@TousifVM:~/sclab/prload$ sudo su
root@TousifVM:/home/tousif/sclab/prload# export LD_PRELOAD=./libmylib.so.1.0.1
root@TousifVM:/home/tousif/sclab/prload# gcc -o myprog myprog.c
root@TousifVM:/home/tousif/sclab/prload# chmod u+s myprog
root@TousifVM:/home/tousif/sclab/prload# ./myprog
Iam not sleeping!
root@TousifVM:/home/tousif/sclab/prload# exit
exit
tousif@TousifVM:~/sclab/prload$
```

•Make myprog a Set-UID user1 program (i.e., the owner is user1, which is another user account), and run it as a different user (not-root user).

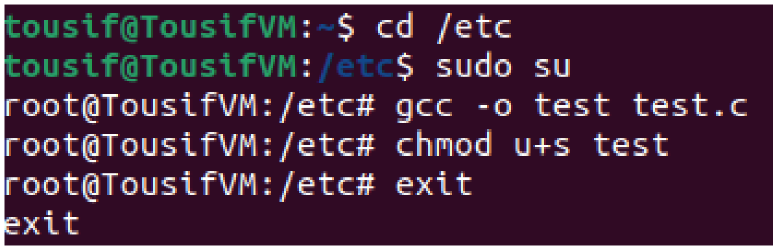


```
user1@TousifVM:/home/tousif/sclab/prload$ export LD_PRELOAD=./libmylib.so.1.0.1
user1@TousifVM:/home/tousif/sclab/prload$ sudo gcc -o myprog myprog.c
[sudo] password for user1:
user1 is not in the sudoers file.  This incident will be reported.
user1@TousifVM:/home/tousif/sclab/prload$ sudo chmod u+s myprog
[sudo] password for user1:
user1 is not in the sudoers file.  This incident will be reported.
user1@TousifVM:/home/tousif/sclab/prload$ su tousif
Password:
tousif@TousifVM:~/sclab/prload$ ./myprog
tousif@TousifVM:~/sclab/prload$
```

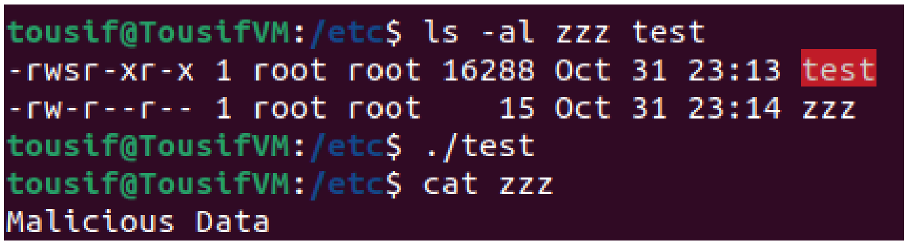
Here, myprog made a Set-UID user1 program and run it as different user.
It will not override sleep() function. Thus, executing sleep() function.

7. Relinquishing privileges and cleanup. To be more secure, Set-UID programs usually call setuid() system call to permanently relinquish their root privileges. However, sometimes, this is not enough. Compile the following program, and make the program a set-root-uid program. Run it in a normal user account, and describe what you have observed. Will the file /etc/zzz be modified? Please explain your observation.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
void main()
{
 int fd;
/*Assume that /etc/zzz is an important system file,*and it is
owned by root with permission 0644.*Before running this
program, you should creat*the file /etc/zzz first.*/
fd = open("/etc/zzz", O_RDWR | O_APPEND);
if (fd == -1)
{
printf("Cannot open /etc/zzz\n");
exit(0);
}
/*Simulate the tasks conducted by the program*/
sleep(1);
/*After the task, the root privileges are no longer needed,it's
time to relinquish the root privileges permanently.*/
setuid(getuid());
/*getuid() returns the real uid*/
if (fork())
{
/*In the parent process*/
close (fd);
exit(0);
}
else
{
/*in the child process*/
/*Now, assume that the child process is compromised,
malicious attackers have injected the following statements
into this process*/
write (fd, "Malicious Data\n", 15);
close (fd);
}
}
```



```
tousif@TousifVM:~$ cd /etc
tousif@TousifVM:/etc$ sudo su
root@TousifVM:/etc# gcc -o test test.c
root@TousifVM:/etc# chmod u+s test
root@TousifVM:/etc# exit
exit
```



```
tousif@TousifVM:/etc$ ls -al zzz test
-rwsr-xr-x 1 root root 16288 Oct 31 23:13 test
-rw-r--r-- 1 root root    15 Oct 31 23:14 zzz
tousif@TousifVM:/etc$ ./test
tousif@TousifVM:/etc$ cat zzz
Malicious Data
```

The file has been modified, because the zzz file is already opened before setuid(). To avoid this, we have to move the line setuid(getuid()) to the front of open() function.