CIS 643 Computer Security

Lab 1 Set-UID Privileged Programs

**Instructor: Wenliang (Kevin) Du**

Contents

1.Set-UID commands 2

2.Set-UID shell 3

(a) Make zsh shell a Set-UID program 3

(b) Make bash shell a Set-UID program 3

3.Setup for the rest of the tasks 4

4.The PATH environment variable 4

(a) Attack under ‘zsh’ shell 4

(b) Attach under ‘bash’ shell 5

5.The difference between system() and execve() 6

(a) Invoke command using system() 6

(b) Invoke command using execve() 7

6.The LD PRELOAD environment variable 7

(a) Make myprog a regular program, and run it as a normal user 7

(b) Make myprog a Set-UID root program, and run it as a normal user 8

(c) Make myprog a Set-UID root program, and run it in the root account 8

(d) Make myprog a Set-UID user1 program, and run it as a different user 8

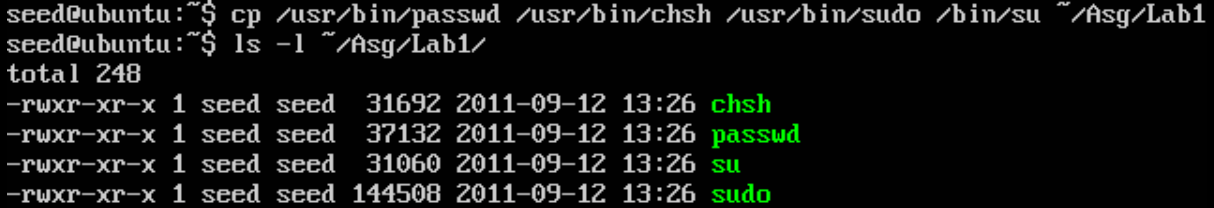
7.Relinquishing privileges and cleanup 9

## 1.Set-UID commands

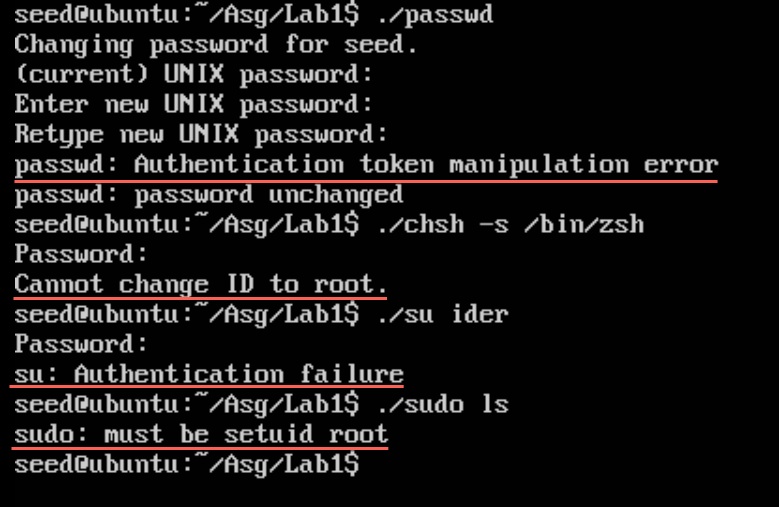
Use ls –l commands, we see that passwd, chsh, sudo, su commands are all owned by root, and the owner permission is rws, which means they are Set-UID programs.



Copy those command programs to normal user’s folder, and we could see the owner is no long root, the commands are neither Set-UID programs.



Run those programs, we could not achieve the results as usual. The reason is that either user has no authentication or user cannot change to root. Because only root user could access these programs or files that need modify.



Event use root to copy those commands to other files, when use normal user to run those program we would still get the same errors as above.

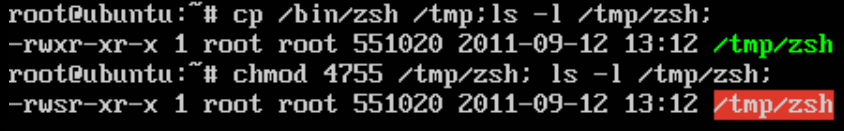


This is what Set-UID works: allow other users to run the programs as the owner.

## 2.Set-UID shell

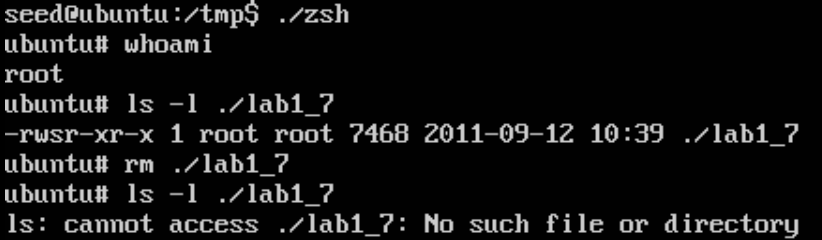
### (a) Make zsh shell a Set-UID program

Use root to copy ‘zsh’ shell to /tmp folder, and set this shell Set-UID program.



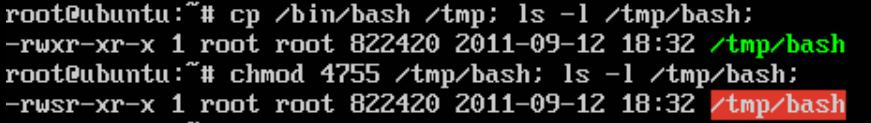
Use seed(a normal user) access to ‘zsh’ shell. Use the “whoami” command, we could see that normal user become root user.

It could also remove the files that owned by root. Which could be further proof of that normal user get root privilege.



### (b) Make bash shell a Set-UID program

Use root to copy ‘bash’ shell to /tmp folder, and set this shell Set-UID program.



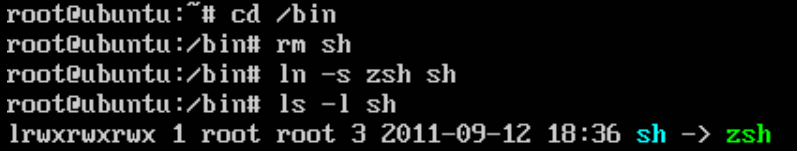
Use seed access to ‘zsh’ shell. Use the “whoami” command, we could see that normal user is still itself.

It could not remove the files that owned by root.



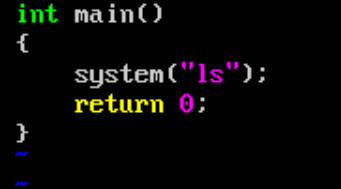
It means bash shell has more limitation on Set-UID, it contains built-in protection to prevent Set-UID mechanism from risk.

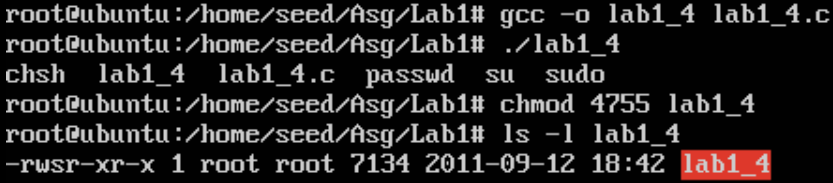
## 3.Setup for the rest of the tasks



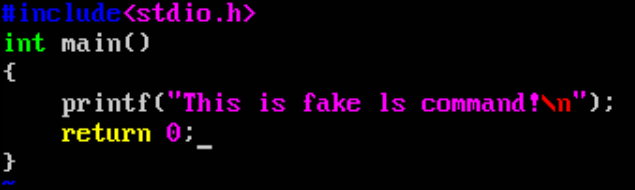
## 4.The PATH environment variable

Login as root, use vim and gcc to create test program.

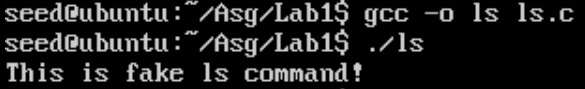




Login as seed, use vim and gcc to create fake ls program.



Run the fake ‘ls’ program, we see the result is just what we put in ls.c file.

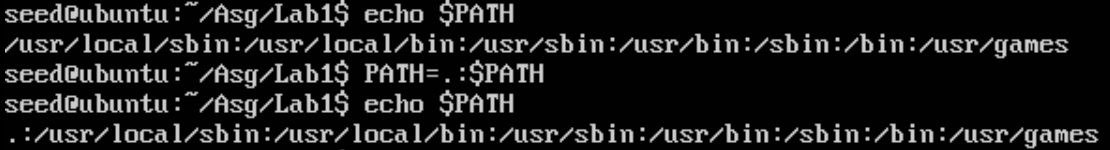


Run the test program that created and owned by root, it is now run the ‘ls’ command in system shell.

Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-09-12 at 11.36.26 PM.png

### (a) Attack under ‘zsh’ shell

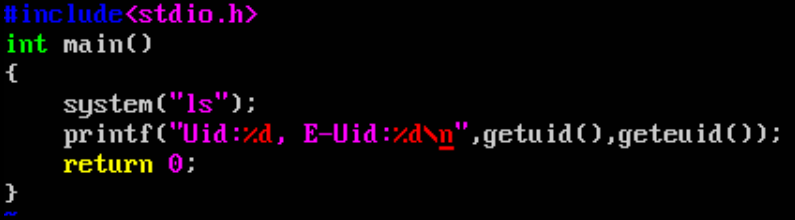
Change the PATH environment variable, add current directory(.) at first place

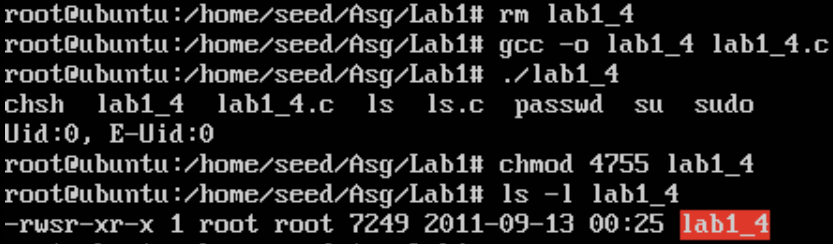


Now run test program again:

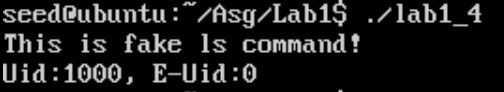
Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-09-13 at 12.02.39 AM.png

Aha, the test program runs the fake ‘ls’ program.

Use root user to modify the test programs, add some output statements for checking whether normal user get root privilege



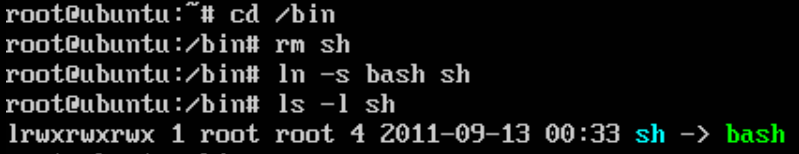
Use normal user to run the test program again



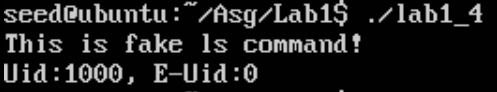
From the result, we can see that the real uid is the user itself, and the effective uid is 0, the uid of root, which proof we have get root privilege.

### (b) Attack under ‘bash’ shell

Change the shell back to ‘bash’



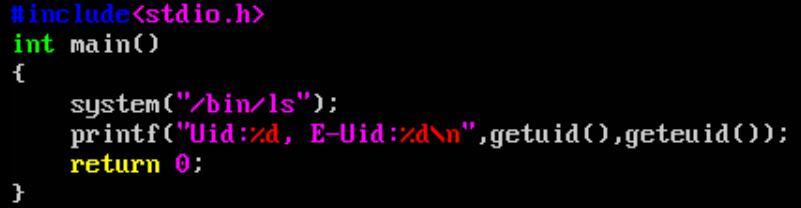
Do the above attack again, finally let normal user run the test program

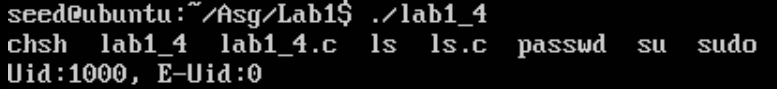


The result is the same as run under ‘zsh’ shell. Which means we can also get root privilege under ‘bash’ shell.

Although ‘bash’ shell could limit abuse of Set-UID programs, but it still could not protect the out of the programs, it could not control the environment where the Set-UID programs are running.

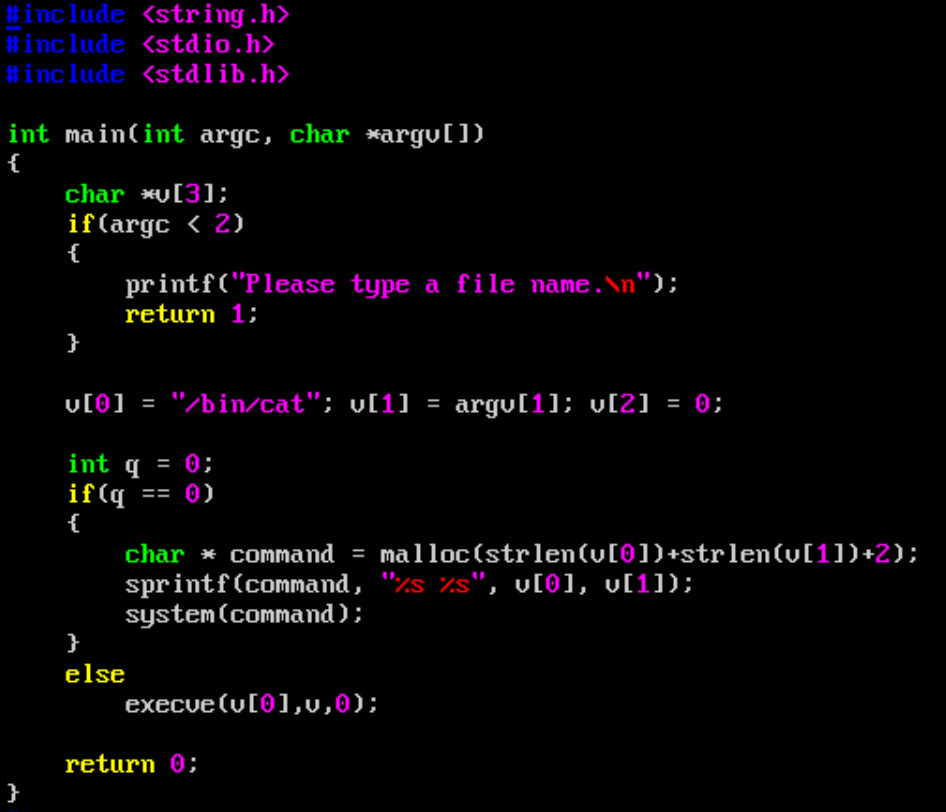
But there are still some ways to protect such attack, e.g. fix the path of the command.



Now when we run the test program, we will get what it really wants to do

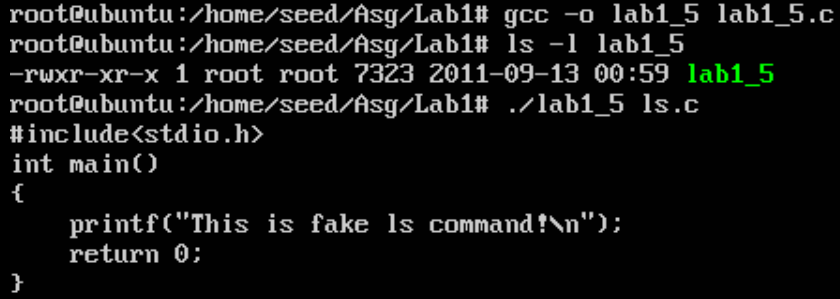
## 5.The difference between system() and execve()

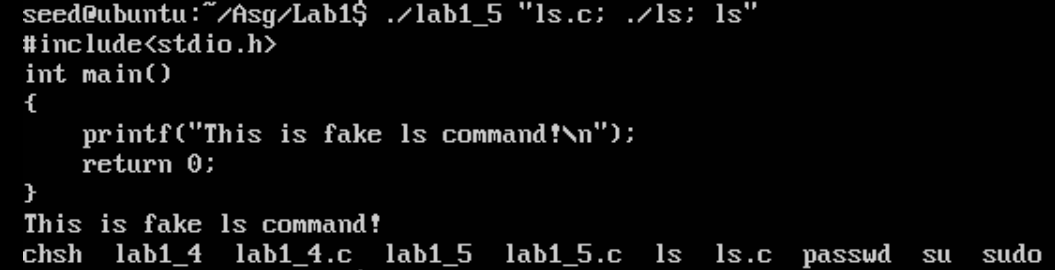
Test program



### (a) Invoke command using system()

Set q to 0, compile the test program, we could see this program has executive permission for group and other, but this program is not a Set-UID program.



Then normal use try to run this program, but input of program, we only give the normal file, but also include commands (use “” to quote strings as one parameter)

We see that the program not only output the content of file, but also run the commands we pass to it.

Even worse, if we pass some commands like “rm /\*” it would destroy the whole system.

### (b) Invoke command using execve()

Set q to 1, compile the test program again, this time the program will use execve() instead of system().

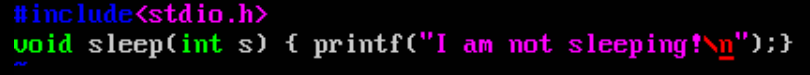
If we pass the same parameter as before

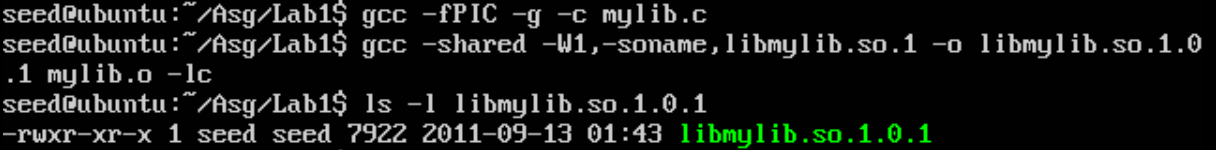
Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-09-13 at 1.22.57 AM.png

The program views the commands in parameter as part of file name. Unfortunately, it could not find the weird file, so it output an error, but it protects system successfully.

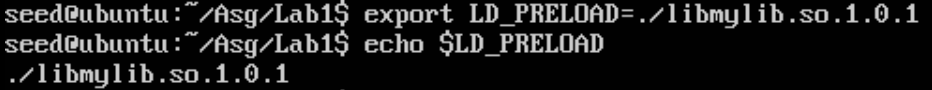
## 6.The LD PRELOAD environment variable

Login as seed, and create the library

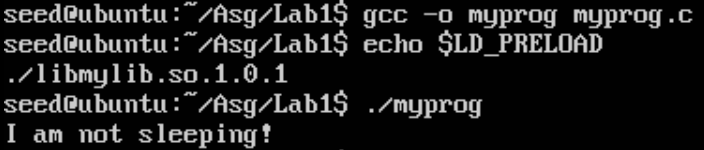




Export the LD\_PRELOAD variable, so that child process could use it



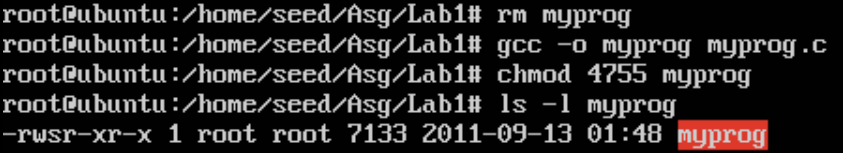
### (a) Make myprog a regular program, and run it as a normal user



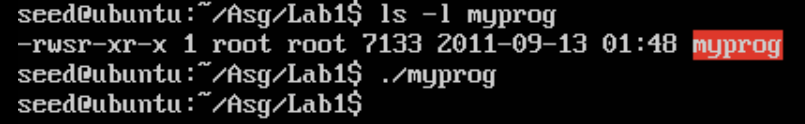
From the output, myprog link to test library instead of system library.

### (b) Make myprog a Set-UID root program, and run it as a normal user

Login as root, and recreate the myprog program, and make it Set-UID program



Run the myprog use seed

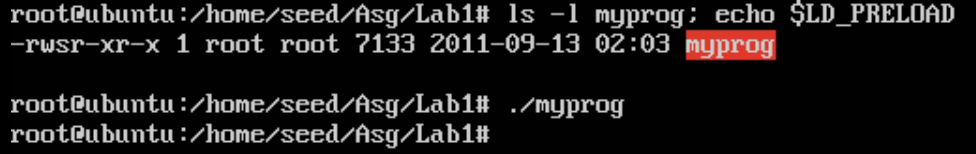


The process stop for a while, and nothing printed on screen.

Which means the test library has been ignored.

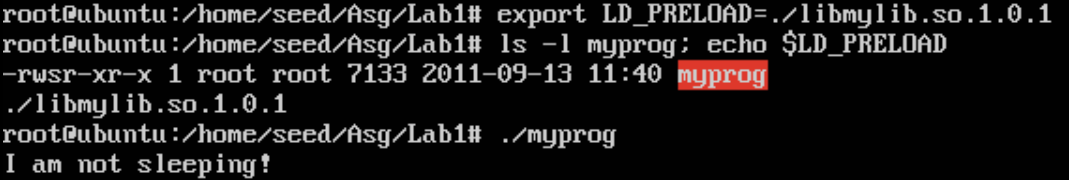
### (c) Make myprog a Set-UID root program, and run it in the root account

Use root user to run myprog



It also ignores the test library.

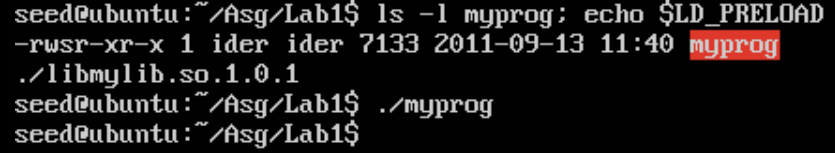
But if we let root export LD\_PRELOAD variable else



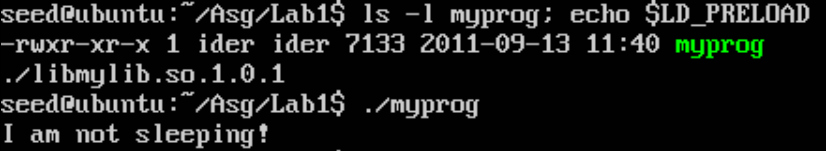
myprog links to test library, though the library is owned by seed.

### (d) Make myprog a Set-UID user1 program, and run it as a different user

Run Set-UID myprog owned by other user



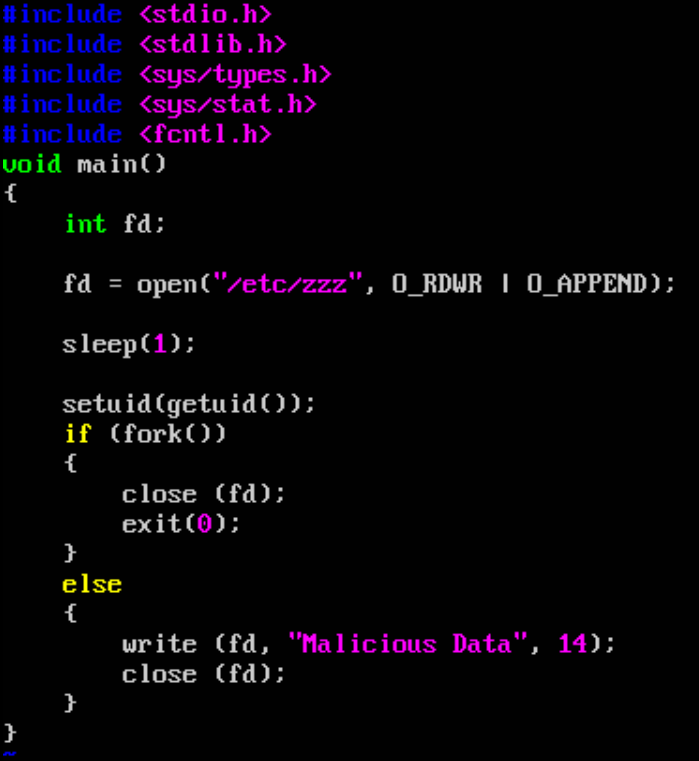
It ignores the test library. But if myprog is not Set-UID program



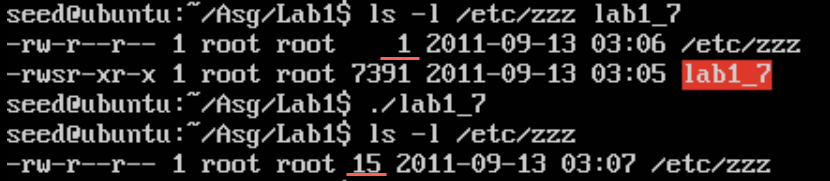
No, the myprog links to the test library.

So, the LD\_PRELOAD variable will lead program to link the specific library instead of standard system library. But if the Set-UID program, system will ignore the variable to make the program safe.

## 7.Relinquishing privileges and cleanup

Test program

Login as seed, run the Set-UID program



Although the program has use setuid() to get off the root status, but from the result, we still modify the read-only file: /etc/zzz

The reason is the program has opened the file before call setuid(). The effective uid is equal to the real uid, but file is opened, normal user still could modify the file. So, we must be very careful to place the setuid().