Set-UID Program Vulnerability Lab 1

Yukui Ye

SUID : 439644268

Question 1:

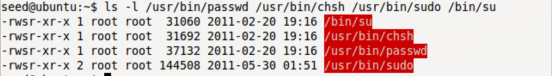
passwd – update user’s authentication

chsh - change your login shell

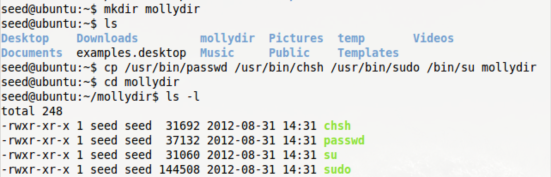
su -run a shell with substitute user and group IDs

sudo -execute a command as another user

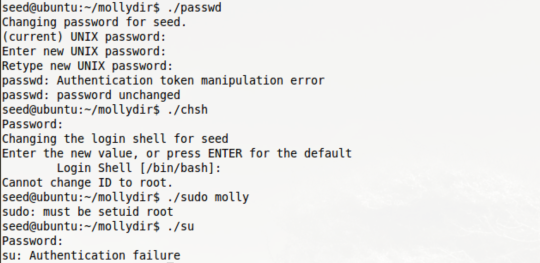
a:First check where is these commands, we found that they are all in Set-UID program.



b:Make my own directory and copy these commands to my own directory, in which we found that the commands in my own directory, they no longer belong to Set-UID.



c: Run these commands. We found that we do not have the right to change it.

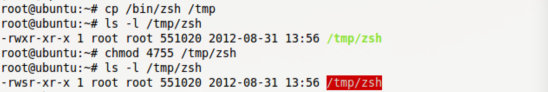


If we change the user to root , and do these commands, we found that it would successfully changed. That is because root is the Set-UID program but seed is not.



Question 2:

a: Copy /bin/zsh to /tmp folder, and make zsh Set-UID program.

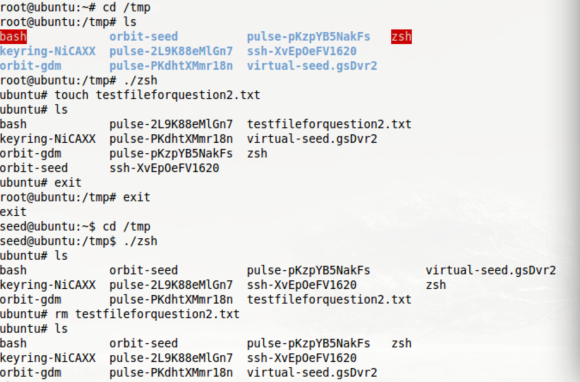


when login as common user and run tmp/zsh, we can tell that normal user become root.



Example for common user still get root privilege by removing file created by root.

First login as root user and creat a testfile called “testfileforquestion2.txt” in the zsh shell. Then login as normal user seed , to delete this testfile by using rm command. We can see from the screenphoto, the testfile has been deleted successfully.



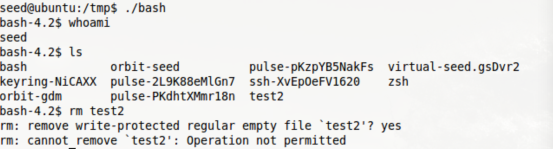
b: Copy /bin/bash to /tmp , and make it Set-UID program



run it as normal user, we found that normal user still be itself, it did not get the privilege of root.



it can not remove the file owned by root.



In sum, Comparing these two attack, bash shell has more limitation on access Set-UID program that shell zsh. You can access root privilege by coping shell zsh to file /tmp. /bin/bash has certain built-in protection that prevent the abuse of the Set-UID mechanism.

Question 3 : Setup for the rest of the task



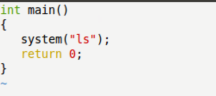
Question 4:

a: login as root creat a test file named “testforQ4.c”by using vi and gcc . and make

the file as Set-UID program.

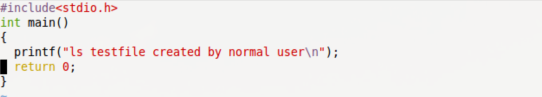


testforQ4.c

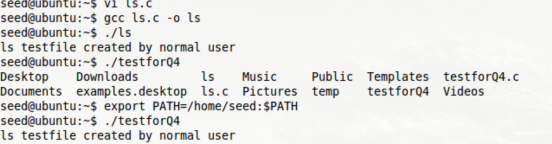


Login as normal user, creat a file named “ls.c” by using vi and gcc.

ls.c



Run ls successfully, then change the path environment, by change the path environment to current path environment, then run file “testforQ4”, we found that “testforQ4”is running the file “ls”.



To prove normal user get the privilege of root :

Edit the file named”testforQ4.c” which original owned by root.



UID is root and EUID is also root.



make testforQ4 file as Set-UID file



Normal user run the file, We can tell that the real uid is the user which is 1000,the effective uid is 0 which is the uid of root; this means normal user has successfully gain the privilege owned by root.



we can also tell that testforQ4.c is now controlled by normal user which originally is Set-UID program.

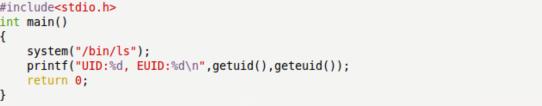
b: change sh shell to bash shell

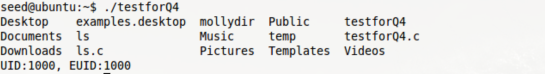


and continue attack it. We got the same result as above. Therefore even if we change the shell from sh to bash, we can still get the same result as above which is we can still have the privilege of root



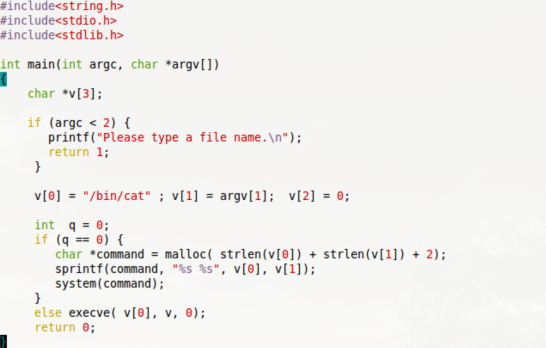
This illustrate that bash shell cannot control environment path and the output of the program although it could limit the abuse access of Set-UID program. but when we change the path of the system cmd from “ls” to “/bin/ls”

We can see that both UID and EUID is all 1000,the normal user itself. That means normal user cannot access this test file anymore.

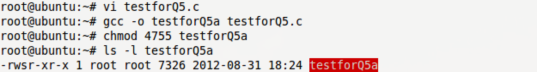


Question5 :

First creat file named “testforQ5.c”



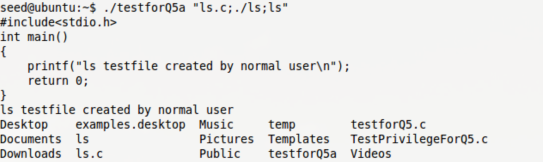
and compile the file and name it as “testforQ5a” and make this program a Set-UID program.



make another Set-UID file called “TestPrivilegeForQ5.c”



then login as normal user to run this program.

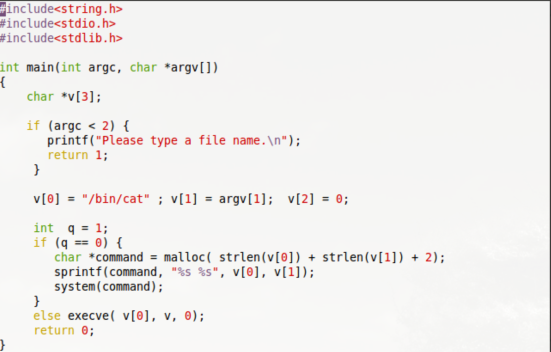


we can see from the above screenshot photo, when you run the program, it first prints out the content of the file and then system call the command which is executing this file. This is danger because you can put any kinds of command in the second input place. Like the following example, you put command”rm TestPrivilegeForQ5.c”, this command means remove this file.



Thus the normal user can remove this file owned by root. By doing the above things, bob can get privilege of the root and therefore can modify the file.

change the q to 1:



compile it and named the file as “testforQ5b”

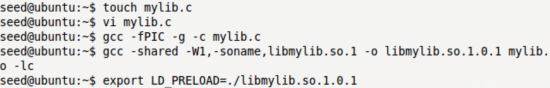


login as normal user

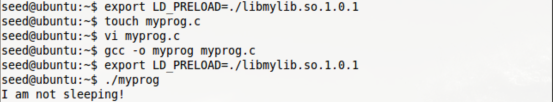


when you change it to execve, the program read the command as part of the file’s name, it did not execute the command at all. Therefore it successfully protect the system by calling execve() instead of system();

Question 6 :

Build a dynamic link library named “mylib.c”, compile it, and set the LD\_PRELOAD environment variable

a: make myprog a regular program, and run it as a normal user. It works well.



b: make myprog a Set-UID program, and run it as a normal user. It stops there , nothing comes out. Even if we let the seed export LD\_PRELOAD variable,it did not work.



c: make myprog a Set-UID program, and run it in the root account. Nothing comes out. The program ignores the command.

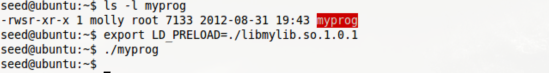


but when we let the root export LD\_PRELOAD, it linked to the specific library “myprog” rather than the standard library.

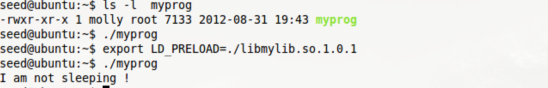


d: make myprog a Set-UID user1 program(the owner is user1,which is another user account),and run it as a different user(not-root user).

Change the owner of the Set-UID myprog program to molly, and run it by login seed account, the program doesn’t work out even exporting the LD\_PRELOAD. Set-UID program protect itself.



when change the program to normal program rather than Set-UID program, it works.



Therefore, the LD\_PRELOAD variable will lead the program to link the specific library which is “mylib” in this example, rather than the standard library

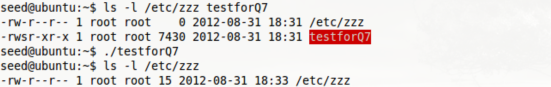
Question 7:

Creat test file “testforQ7”and file “/etz/zzz”

make /etc/zzz permission as 0644 and make test file as Set-UID program



login as normal user, and run the test file. After running the test program, the /etc/zzz file number changed from 0 to 15.



The program use setuid() to relinquish the root privilege, that is when calling setuid(getuid) function, it makes all Set-UID program to get their real user ID (getuid) rather than effective id(geteuid). but we can still modify the read-only file etc/zzz due to the program has already opened the file before calling setuid().therefore the normal user can modify the file.