Lab3 : Race Condition Vulnerability Lab

Yukui Ye

SUID : 439644268

*Initial setup: First turn off the protection.*



*Task1: Exploit The Race Condition Vulnerability*

Step1:The Vulnerable Program Named ‘vulp.c’



Compile it and make it as Set-UID program



Step2:Creat a file named “addfile” under /tmp , which store a new user,encrypted password and with root privilege uid =0;



then Creat the shell script “check.sh”which indicate whether the /etc/shadow has been modified. It will prints a message if the time stamps of /etc/shadow has been changed .



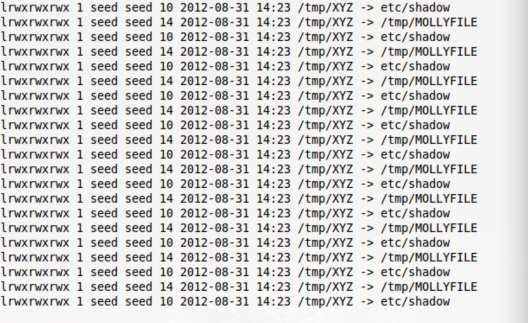
Step3: Creat the attact program named”attackprog.c”



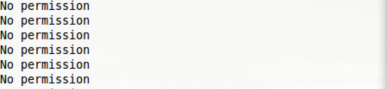
Compile it to a file named ‘attackprog’



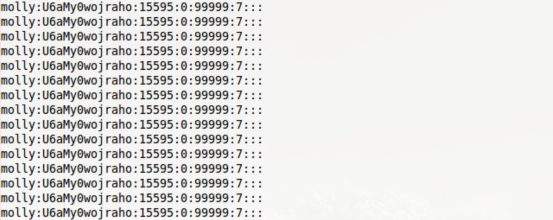
Step4: Start to attack: Run shell script and attack program in different terminal;

The output of the attack program :

The output of the shell script :



During the attack procedure, the context would also write into MOLLYFILE. Check the output of the MOLLYFILE:



Attack successed, the check.sh stopped



Login root to check /etc/shadow.



Step5:use the same method to attack /etc/passwd. We will successfully add out user”molly” which owned the root privilege in /etc/passwd. and then switch user to molly which use this command “su molly”, at last type “whoami” command. It clearly shows root.



Task2: Protection Mechanism A : Repeating

Step1: Change the attackprog.c to the following code, in order to add multiple check to this program. In this code, “lstate()” means to check i-node information, then use access to check permission more multiple times.



Step2:Compile this file named “vulp2”



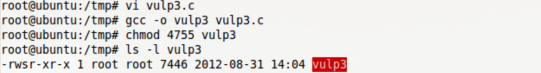
Step3: Do attack as before. Waiting for almost one hour, it did not succeed. but we do write a lot of text in MOLLYFILE which owned by seed. Anyway, this protection mechanism worked since we could not write file owned by root.

Task3: Protection Mechanism B: Principle Of Least Privilege

In this task, we introduce a new protection mechanism which main idea is the transaction between uid and euid, though function seteuid(); The changed code is as following.



Make it as Set-UID program

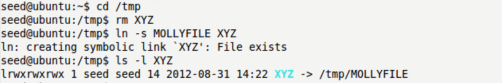


Do the attack as before, the program can only write text into MOLLYFILE, It cannot access either /etc/shadow or /etc/passwd. This is because open()function only check whether the effective user have permission to open file to write or read, getuid() aimed to get real user id, while seteuid( getuid() ) make the effective user id to the real user id, therefore it has no permission to access root file.

Task4: Protection Mechanism C: Ubuntu’s Built-In Scheme

Step1: Unlock the protection, set the value from 0 to 1;



Step2:Link file “XYZ” to “MOLLYFILE”.

Step3: Run the vulp program.



*Question1: why does this protection scheme work ?*

Answer: symlinks in world-writable sticky directories such as /tmp cannot be accessed if the follower and directory owner do not match the symlink owner. Since /etc/passwd and /etc/shadow are owned by root, even when user make a link from XYZ to these file, the scheme does not allow to access.

*Question2: Is this a good protection ? why or why not?*

This protection is just okay, Normally, when we make a link from XYZ to MOLLYFILE which owned by seed, this program should allow us to access the file and write text to aimed file, but in task 4, we only get “No permission” which means we are not allowed to write the file owned by the same user.

*Question3: What are the limitations of this scheme ?*

This scheme limit user to open file through link command. It protects files owned by root.