CIS 643 Computer Security

Lab 3 Race Condition Vulnerability Lab

**Instructor: Wenliang (Kevin) Du**

Contents

Task 1: Exploit the Race Condition Vulnerabilities 2

Vulnerable Program 2

Attack Program 3

Begin Attack 3

Task 2: Protection Mechanism A: Repeating 5

Task 3: Protection Mechanism B: Principle of Least Privilege 5

Task 4: Protection Mechanism C: Ubuntu’s Built-in Scheme 6

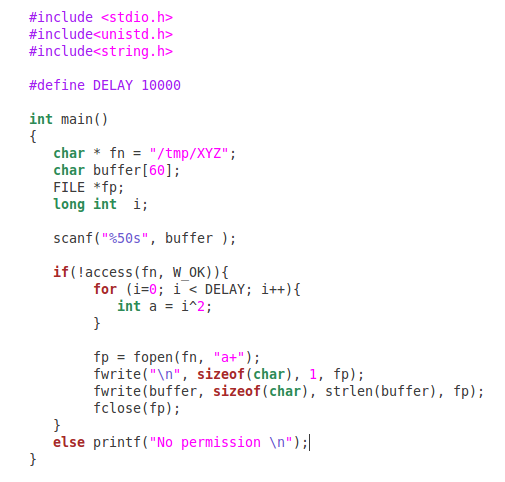
(1) Why does this protection scheme work? 7

(2) Is this a good protection? Why or why not? 7

(3) What are the limitations of this scheme? 7

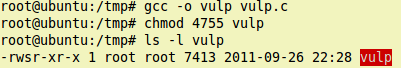
## Task 1: Exploit the Race Condition Vulnerabilities

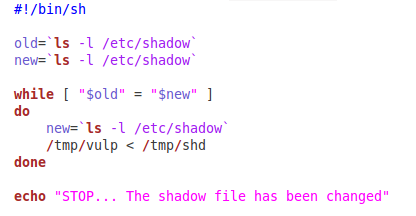
### Vulnerable Program



The program use access() to check is real user has permission to access the file, if so it open the file, and append text to the file.

Use root user to compile the program and set it Set-UID program.



Use shell script to run the vulnerable program many times, and check if the target file(/etc/shadow) has been modified: 

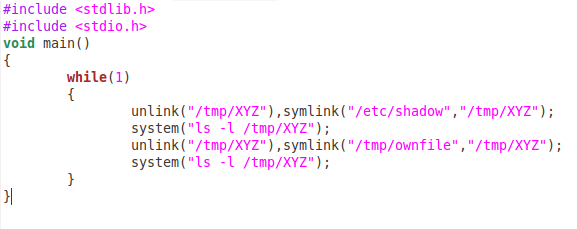
The “shd” file contains user information, which the UID is 0

Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-09-26 at 9.16.01 PM.png

Turn off the Ubuntu image build-in protection:

Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-09-26 at 10.23.54 PM.png

### Attack Program



This program just make a link file and switch linking to /etc/shadow file and /tmp/ownfile (owned by seed) file.

It also check which file do XYZ current link to.

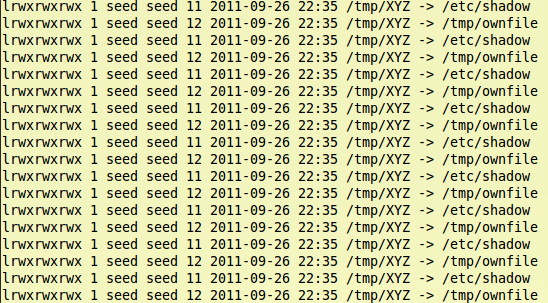
Compile the attack file:

Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-09-26 at 10.29.02 PM.png

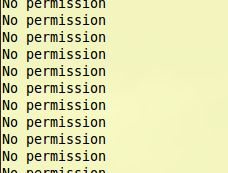
### Begin Attack

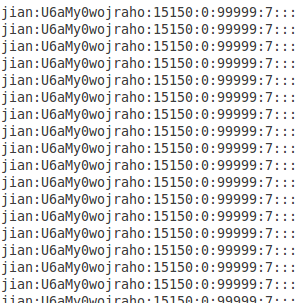
Run shell script and attack program in different terminal window

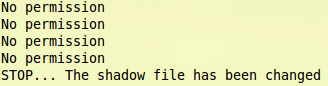
The attack program will output the link information:



The vulnerable program, which is running in shell script, just output “No permission”:



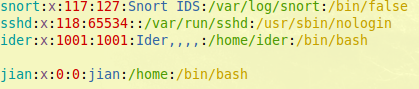
During this progress, some times text will be written into ownfile:

Once attack succeed, the shell script will stop

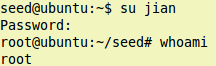
Use root to check the /etc/shadow file, we will see that our user has been add to shadow file:



Use the same steps to attack /etc/passwd file, finally we also add user info to it:



If we switch to the ‘jian’ user, we could see that we get the root permission without nay password:



## Task 2: Protection Mechanism A: Repeating

This time, we change the vulnerable program, adding multiple checks to program:

use lstate() to check i-node information;

use access() to check permission more multiple times.



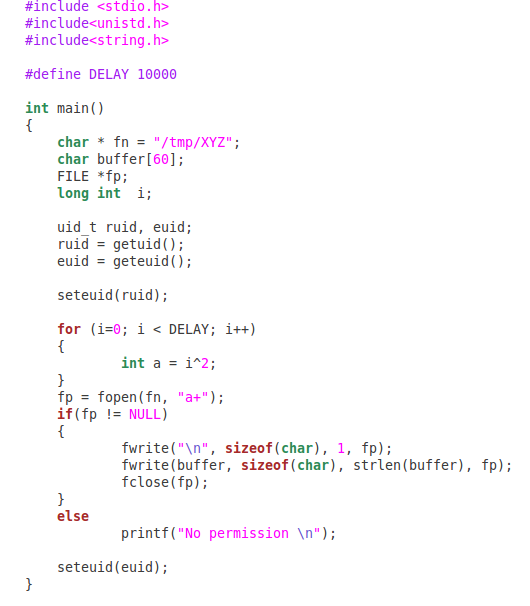
Do attack steps as before again.

After long time(about 40 minutes), we still could not write protected root files, we write a lot text to “ownfile”, which is owned by seed.

So we could say that the protection scheme works.

## Task 3: Protection Mechanism B: Principle of Least Privilege

This time, we use seteuid() method in Set-UID vulnerable program to make the effective user equal to real user, after open and write the text to file, change it back:



As the protection mechanism A, this protection also prevents the root files from vulnerable program to write.

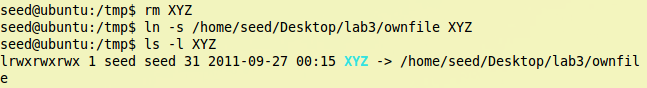
The program only write the text into “ownfile”, but no way to access /etc/shadow and /etc/passwd, as open() function will check if the effective user have permission to open files for writing or reading. Unfortunately, seteuid() make the effective uid to be the same as real uid, which is no permission to the root files.

## Task 4: Protection Mechanism C: Ubuntu’s Built-in Scheme

Change the vulnerable program as it in Task 1.

Turn on the Ubuntu build-in sticky symlinks.

Macintosh HD:Users:Ider:Desktop:Screen Shot 2011-09-27 at 12.05.50 AM.pngUse command to link XYZ to “ownfile”. This time, the file is in different directory



Run the vulnerable program on terminalMacintosh HD:Users:Ider:Desktop:Screen Shot 2011-09-27 at 12.16.18 AM.png

We got “Segmentation fault” directly, we even could not run the program.

### Why does this protection scheme work?

In Ubuntu 10.10 and later, symlinks in world-writable sticky directories (e.g. /tmp) cannot be followed if the follower and directory owner do not match the symlink owner[[1]](#footnote-1).

The /etc/shadow and /etc/passwd are owned by root, so when seed make XYZ link to those file, the scheme not allow to access and even write to these files.

### Is this a good protection? Why or why not?

The protection works well, but as I tell, the protection is not so good.

In Task 4, I just make XYZ link to the file that owned by seed but in different places, in this case, the program should allow user to open the link and write text to the real file.

Unfortunately from the Task 4, we only get “Segment Fault” error, which means we are also not allowed to write files owned by the same user.

### What are the limitations of this scheme?

The scheme protects the root files, but also limit user to open files through links. However the scheme is really safe.

1. Reference from <https://wiki.ubuntu.com/Security/Features#Symlink_restrictions> [↑](#footnote-ref-1)