HackerFrogs Afterschool Network Hacking – Session 10

Class:
Network Hacking

Workshop Number:
AS-NET-10

Document Version:
1.75

Special Requirements:
Registered account
at tryhackme.com

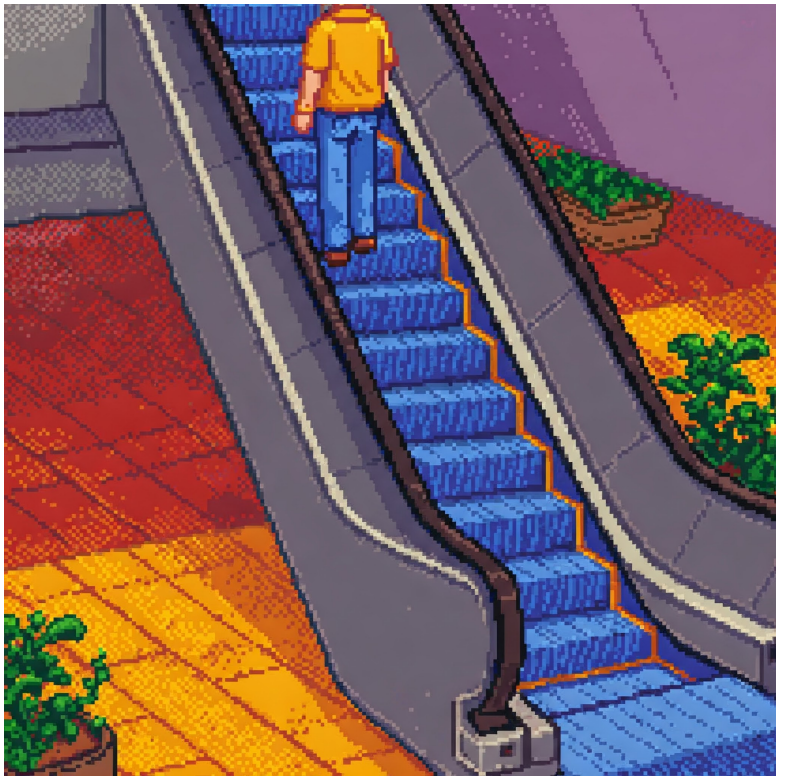
Welcome to HackerFrogs Afterschool!

This is the tenth session for network hacking!

Let's go over the concepts we covered in the previous session!

What is Privilege Escalation?

Privilege escalation (priv esc), is the act of upgrading the level of access on a system, typically through abusing insecure settings or capturing credentials



Priv Esc via Sudo

```
User theshyhat may run the following commands
    (ALL : ALL) ALL
```

If a user has access sudo (root) access with any command, there is the possibility of abusing that command for privilege escalation

Priv Esc via SUID

```
└─$ ls -la /usr/bin/chfn
-rwsr-xr-x 1 root root 70888 Aug  5  2024 /usr/bin/chfn
```

SUID binaries are commands that are run in the context of the file's owner by default. Most SUID binaries are owned by the **root** user, and certain SUID binaries can be used for privilege escalation

Priv Esc via Kernel Exploits

Kernel exploits are hacks that target the lowest level of operations in an operating system

Typically, older versions of operating systems are vulnerable to kernel exploits

This Session's Topics

- Priv Esc via PATH Hijacking

- Priv Esc via Capabilities

- Priv Esc via Cron Jobs

Accessing TryHackMe

Let's access this TryHackMe room to learn about privilege escalation:

https://tryhackme.com/room/linprivesc

Priv Esc via PATH Hijacking

```
alper@targetsystem:~/Desktop$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
alper@targetsystem:~/Desktop$
```

In Linux, the PATH variable is a list of directories where the system will look for command binaries

Priv Esc via PATH Hijacking

```
alper@targetsystem:~/Desktop$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
alper@targetsystem:~/Desktop$
```

For example, if you use the **whoami** command, Linux will look in the /usr/local/sbin directory for the command first, then the /usr/local/bin directory, etc, etc, until it gets to the end of the PATH directories

Priv Esc via PATH Hijacking

```
#!/bin/bash

echo "The current user is $(whoami)"
```

If we locate a privileged script, a command we can run with sudo permissions or a SUID binary that is using another binary **without an explicit file path**...

Priv Esc via PATH Hijacking

```
export PATH=/tmp:$PATH
```

Then we could add a writable directory to the path...

Priv Esc via PATH Hijacking

```
echo "/bin/bash" > /tmp/whoami
```

Then create a malicious version of whatever command is being referenced in the privileged binary / script

Priv Esc via PATH Hijacking

Now, when we run the privileged file, it will open up a privileged shell

Priv Esc via Capabilities

```
$ getcap -r / 2>/dev/null
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstrea
 = cap_net_bind_service,cap_net_admin+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/bin/ping = cap_net_raw+ep
/home/karen/vim = cap_setuid+ep
/home/ubuntu/view = cap_setuid+ep
```

Capabilities in Linux are a feature which allows for certain binaries / commands to operate with additional privileges

Priv Esc via Capabilities

```
$ getcap -r / 2>/dev/null
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstrea
 = cap_net_bind_service,cap_net_admin+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/bin/ping = cap_net_raw+ep
/home/karen/vim = cap_setuid+ep
/home/ubuntu/view = cap_setuid+ep
```

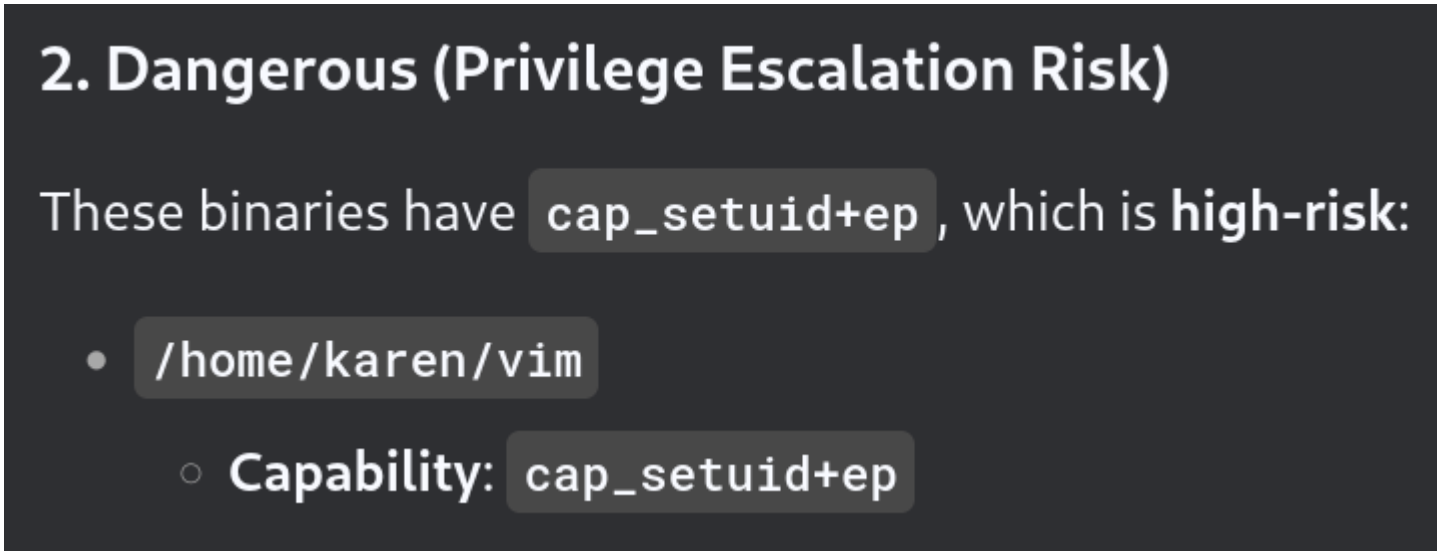
As opposed to SUID binaries or Sudo permissions, Capabilities allow granular control over binary controls

Priv Esc via Capabilities

```
$ getcap -r / 2>/dev/null
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstrea
 = cap_net_bind_service,cap_net_admin+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/bin/ping = cap_net_raw+ep
/home/karen/vim = cap_setuid+ep
/home/ubuntu/view = cap_setuid+ep
```

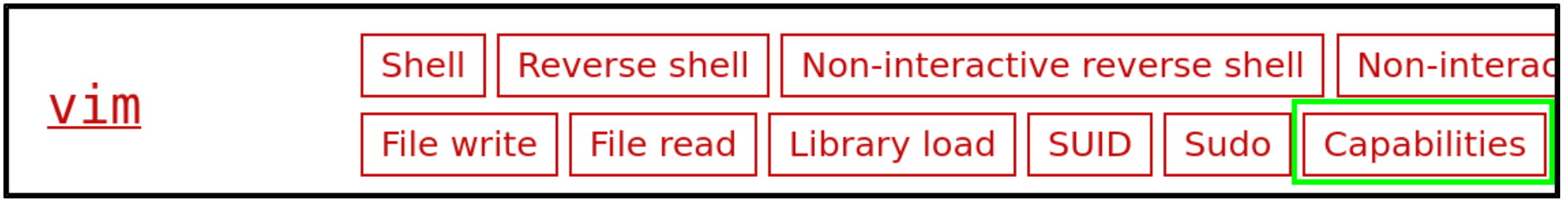
We can return binaries with modified capabilities by using the **getcap** command

Priv Esc via Capabilities



We can feed the list of capabilities binaries to an AI to discover if any of them are potentially vulnerable

Priv Esc via Capabilities



Once a potential vulnerable binary is found, we can check how to use it for priv esc using the GTFOBins website

Priv Esc via Cron Jobs

```
$ cat /etc/crontab
```

```
* * * * *   root /antivirus.sh
* * * * *   root antivirus.sh
* * * * *   root /home/karen/backup.sh
* * * * *   root /tmp/test.py
```

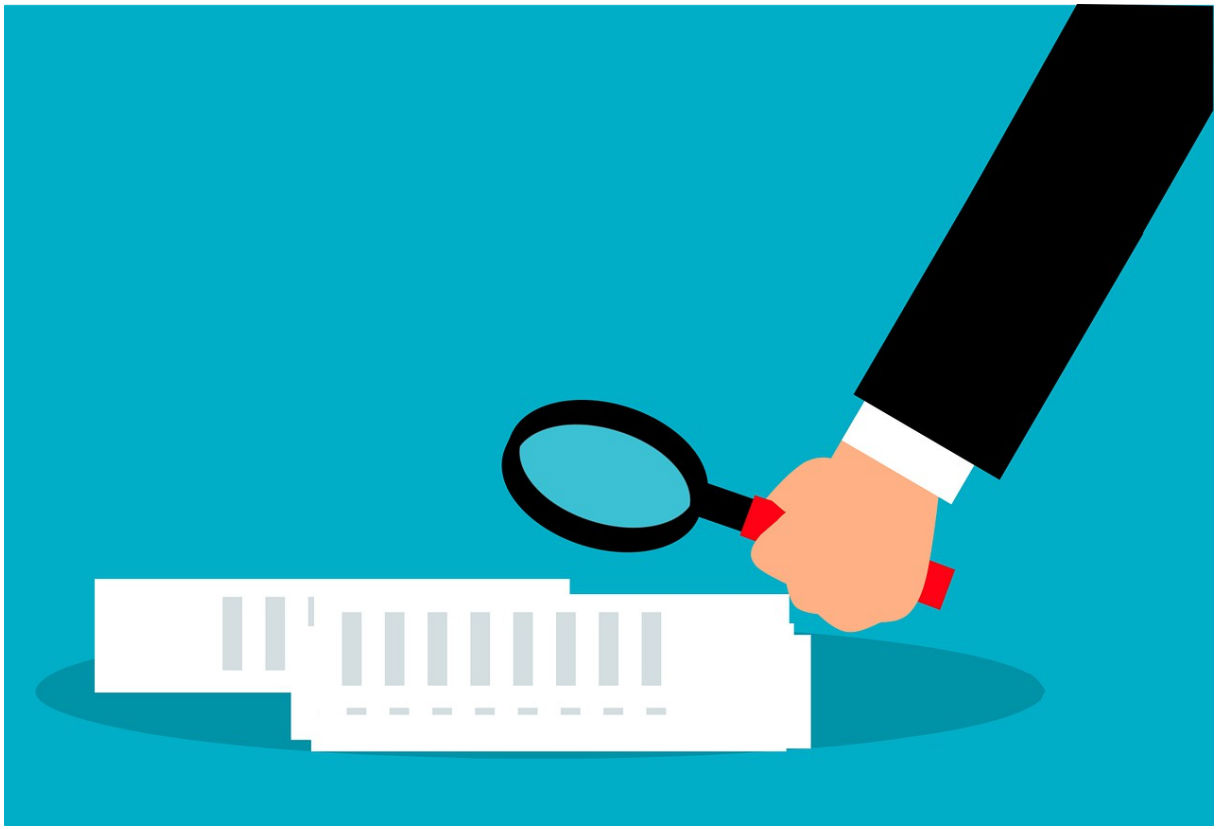
Cron Jobs are a feature in Linux which allows certain binaries or scripts to run at regular intervals (once a minute, once an hour, etc, etc)

Priv Esc via Cron Jobs

```
$ whoami
karen
$ ls -la /home/karen/backup.sh
-rw-r--r-- 1 karen karen 77 Jun 20  2021 /home/karen/backup.sh
```

If we are able to modify a script that is being used for a cron job, or if we are able to modify a component the cron job command / script uses, then we can potentially escalate our privileges

Summary



Let's review the network hacking concepts we learned in this workshop:

Priv Esc via PATH Hijacking

```
alper@targetsystem:~/Desktop$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
alper@targetsystem:~/Desktop$
```

In Linux, the PATH variable is a list of directories where the system will look for command binaries

Priv Esc via PATH Hijacking

```
alper@targetsystem:~/Desktop$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
alper@targetsystem:~/Desktop$
```

A system's PATH can be used as a vector for priv esc if privileged programs use other commands without explicit filepaths

Priv Esc via Capabilities

```
$ getcap -r / 2>/dev/null
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstrea
 = cap_net_bind_service,cap_net_admin+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/bin/ping = cap_net_raw+ep
/home/karen/vim = cap_setuid+ep
/home/ubuntu/view = cap_setuid+ep
```

Capabilities in Linux are a feature which allows for certain binaries / commands to operate with additional privileges

Priv Esc via Capabilities

```
$ getcap -r / 2>/dev/null
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstrea
 = cap_net_bind_service,cap_net_admin+ep
/usr/bin/traceroute6.iputils = cap_net_raw+ep
/usr/bin/mtr-packet = cap_net_raw+ep
/usr/bin/ping = cap_net_raw+ep
/home/karen/vim = cap_setuid+ep
/home/ubuntu/view = cap_setuid+ep
```

Capabilities can be used for priv esc if the capabilities are given to certain binaries / commands

Priv Esc via Cron Jobs

```
$ cat /etc/crontab
```

```
* * * * *   root /antivirus.sh
* * * * *   root antivirus.sh
* * * * *   root /home/karen/backup.sh
* * * * *   root /tmp/test.py
```

Cron Jobs are a feature in Linux which allows certain binaries or scripts to run at regular intervals (once a minute, once an hour, etc, etc)

Priv Esc via Cron Jobs

```
$ whoami
karen
$ ls -la /home/karen/backup.sh
-rw-r--r-- 1 karen karen 77 Jun 20  2021 /home/karen/backup.sh
```

If we are able to modify a script that is being used for a cron job, or if we are able to modify a component the cron job command / script uses, then we can potentially escalate our privileges

What's Next?

The HackerFrogs Afterschool Network Hacking course is done!

But tune in next time when we'll be looking at a new topic: Reverse Engineering!